



UNIVERSITY OF STIRLING

DEPARTMENT OF COMPUTER SCIENCE &  
MATHEMATICS  
DOCTORATE OF PHILOSOPHY

---

# Explaining solutions for decision support via Partial Solutions

---

*Supervised by:*

Dr. Alexander Brownlee  
Dr. David Cairns

*Submitted by:*

GianCarlo A.P.I. Catalano

June 2026



## DECLARATION

---

I hereby declare that this dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text and bibliography.

I also declare that this dissertation (or any significant part of my dissertation) is not substantially the same as any that I have submitted, or that is being concurrently submitted, for a degree or diploma or other qualification at the University of Stirling or similar institution.

I was admitted as a research student in June 2023 (for a Masters of Philosophy) and then accepted as a candidate for the degree of Doctor of Philosophy in 2025. This dissertation is a record of the work carried out at the University of Stirling between 2023 and 2026 under the supervision of Dr Alexander Brownlee, Dr David Cairns, Prof John McCall and Dr Russell Ainslie.

I have read, and adhered to, the University's plagiarism policy.

*Stirling, June 2026*

---

GianCarlo A.P.I. Catalano

## ABSTRACT

---

Optimisation problems arise across numerous domains, with **metaheuristics** being widely adopted due to their flexibility and ease of use. Although the metaheuristic framework is relatively simple, the search process is sufficiently complex that it is effectively a black box. Users are often forced to accept results at face value despite the risks of faulty implementations, non-determinism, and unintended biases. To address these limitations, this research introduces an **explainability** framework for metaheuristics by extracting human-interpretable insights from generated solutions.

Central to the proposed methodology is the novel concept of **Partial Solutions**—defined and implemented as part of this work—which represent understandable sub-structures within metaheuristic solutions associated with high fitness. To find these patterns, we use a **multi-objective search method** that leverages information relating to their complexity, fitness, and structure. The resulting patterns offer trade-offs between objectives, and each can be post-processed into an explanation.

To quantify the “structure” of Partial Solutions, we developed two novel **linkage learning** methods and a new linkage evaluation procedure, which we show offers a superior trade-off between accuracy and execution time compared with existing methods.

The utility and domain-agnostic nature of this framework are demonstrated and evaluated across several distinct applications. First, the methodology was applied to a complex industrial optimisation problem posed by British Telecom (BT), yielding insightful explanations. Second, the framework was adapted for a **recommender system** model, successfully generating explanations that satisfy stakeholder objectives. Furthermore, the same underlying methodology was utilised to construct **transparent models for predicting solution fitness**.

To conclude, Partial Solutions are a flexible tool for explainability. They are shown to recover known patterns within benchmark problems, and to be applicable and insightful in large and complex problems such as British Telecom’s Staff Rostering. Our methodology can also be used beyond metaheuristics; for example, we apply it to a recommender system to produce domain-specific explanations. Finally, Partial Solutions can be used to predict fitness: we show that decision trees based on Partial Solutions are competitive with traditional approaches.

We conclude the thesis with reflections on whether our objectives have been achieved, and possible future work. Partial Solutions may be used to implement efficient and more understandable metaheuristics, and the applicability of the explanations produced may be expanded beyond combinatorial optimisation.

## ACKNOWLEDGMENTS

---

My work is thanks to the support of many people. Financially, my work is supported by the DataLab and BT Research. Academically, I have been supervised by:

- **Alexander Brownlee** and **David Cairns** (from the University of Stirling),
- **John McCall** and **Martin Fyvie** (from the National Subsea Centre)
- **Russell Ainslie** (from British Telecom Research)

I am deeply grateful for my 2-year sponsorship granted by BT, and for my supervisors who have supported me and allowed me to convert my MPhil into a PhD. Sandy has sacrificed much of his time and effort for me (as well as some grant money to let me attend conferences!), and I hope I can repay the favour one day. I have been supervised by David for 4 years (counting my undergrad), and I have always enjoyed our (not always on topic) meetings.

I have also collaborated with Piotr Lipiński and Klaudia Dynak (née Balcer) from the **University of Wrocław** when working on [43], I am eternally grateful for allowing me to collaborate with them.

Additionally, my work was supported through other forms of funding:

- **Dr. Saemundur Haraldsson**, who through Teaching Assistance positions was able to extend my research project
- SymphoMe (via Ifeanyichukwu Ezimandu, Yves Segovia and supervised by **Jason Adair**) who funded a Research Assistant position for me to extend my funding
- **Prof. Rachael Norman**, who hired me as a **Research Assistant** position for the Scottish Alliance for Food

I have read many PhD thesis acknowledgements, and I feel compelled to give advice to PhD students reading this. PhD programs are always different (the person doing them, the supervisors, the topic, the time available, the state of the economy...), and most of the advice you hear is not universally applicable. The saying of “not a sprint, rather a marathon” is probably the most general and important advice I can give; I am exhausted now and I should have rested more. A distraction paired with passion - a partner and a hobby especially - go a long way in forcing you to relax. As for practical advice, I would suggest setting `typehere.cc` as your new tab in your browser to write quick notes and to-do lists.

The help I received has gone beyond financial and academic, and I would like to thank my family, my partner Joe and my friends (especially Molly and Archie). I am grateful for the distractions you have provided. The other friends from 4B101 - Stefano, Penny, Osamu, Jakub, Craig and the *many* visiting PhDs - have also assisted in the impossible endeavour of distracting me... thank you! Stefano is the reason why I am currently employed as a teacher in the International Study Centre, and I am very thankful for all the staff who have accepted me there as one of their own.

Many other people have helped me throughout this journey, and I hope I have provided each of them with an origami. The following table includes all the ones I could think of.

| Person               | Origami                 | Original Author     |
|----------------------|-------------------------|---------------------|
| Alexander Brownlee   | Giraffe                 | Gen Hagiwara        |
| Alexander Carmichael | Vulture                 | Ho Van Triet        |
| Andrew Hoyle         | Gerbil/Jerboa           | Tran Trung Hieu     |
| Ashley McClenaghan   | Sleeping Cat            | Katsuta Kyohei      |
| Badarrudin Chachar   | Duckling                | Katsuta Kyohei      |
| Craig Reid           | Ragdoll Cat             | Katsuta Kyohei      |
| David Cairns         | Giraffe                 | Gen Hagiwara        |
| Gabriela Ochoa       | Dolphin                 | Robert J. Lang      |
| Gavin Abernethy      | Triceratops             | Fumihaki Kawahata   |
| Grace McArthur       | Giraffe                 | Seth Friedman       |
| Hiba Ahmed           | Pegasus                 | Robert J. Lang      |
| Hicham Ourchani      | Squirrel                | Beth Johnston       |
| Jakub Konopka        | Fox                     | Saku B              |
| Jason Adair          | Ring Tailed Lemur       | Roman Diaz          |
| Joe McLellan         | Golden Molly            | Robert J. Lang      |
| John McCall          | HexaFlexagon            | Author Unknown      |
| Josip Hrvatic        | Highland Cow            | Quentin Trollip     |
| Justin Scouarnec     | Elk                     | Yasushi Miyashita   |
| Klaudia Balcer       | Sheep                   | Beth Johnson        |
| Kevin Swingler       | Sailfish                | Robert J. Lang      |
| Magda Smolic-Rocak   | Panda                   | Yoo Tae Yong        |
| Martin Fyvie         | Frill-Necked-Lizard     | Gen Hagiwara        |
| Noor Ul Zain         | Sleeping Cat            | Katsuta Kyohei      |
| Osamu Takahashi      | Standing Cat            | Katsuta Kyohei      |
| Paul McMenemy        | Dodecahedron            | Ekaterina Lukasheva |
| Penny Johnston       | Squirrel                | Beth Johnson        |
| Piotr Lipinski       | Wolf                    | Weesel Origami      |
| QinBo Yang           | Panda                   | Bernie Peyton       |
| Rachael Normal       | Blue Rhino              | Quentin Trollip     |
| Reyhan Ahmadi        | Squirrel                | Beth Johnson        |
| Robin Hankin         | Rhimbic Triacontahedron | Giancarlo Catalano  |
| Russell Ainslie      | Flasher                 | Jeremy Shafer       |
| Sahar Ebrahimi       | Sleeping Cat            | Katsuta Kyohei      |
| Sarah Thompson       | Shark                   | Michael LaFosse     |
| Simona Hapca         | Roses                   | Naomiki Sato        |
| Stefano Sarti        | Bear                    | Stephen Weiss       |
| Wen Shin Lee         | Sleeping Cat            | Katsuta Kyohei      |

## LIST OF PUBLICATIONS

---

Parts of the work presented in this thesis appear in the following publications:

1. **GianCarlo A.P.I. Catalano**, Alexander E. I. Brownlee, David Cairns, John McCall, and Russell Ainslie. “Mining Potentially Explanatory Patterns via Partial Solutions.” In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 567–570. 2024. [44]
  - Included in Chapter A is an extended 9-page version, with additional information about the evaluation.
2. **GianCarlo A.P.I. Catalano**, Alexander E. I. Brownlee, David Cairns, John A. W. McCall, Martin Fyvie, and Russell Ainslie. “Explaining a Staff Rostering Problem Using Partial Solutions.” In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pp. 179–193. Springer, 2024. [46]
3. **GianCarlo A.P.I. Catalano**, Alexander Brownlee, David Cairns, Russell Ainslie, and John McCall. “Interpretable Decision Trees to Predict Solution Fitness.” In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1099–1107. 2025. [45]
4. **GianCarlo A.P.I. Catalano**, Alexander E. I. Brownlee, David Cairns, John McCall, and Russell Ainslie. “Evaluating a Novel Explainability Method for Metaheuristics via a User Study.” In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1984–1991. Association for Computing Machinery, 2025. [42]
5. **GianCarlo A.P.I. Catalano**, Klaudia Dynak, Alexander E. I. Brownlee, and Piotr Lipinski. “Explaining Recommender Systems’ Performance via User Behaviour Patterns.” In *Artificial Intelligence XLII*, pp. 35–47. Springer Nature Switzerland, 2026. [43]

## NOTATION

---

|                               |                                                      |
|-------------------------------|------------------------------------------------------|
| $L$                           | Number of decision variables in the search space.    |
| $D_i$                         | Domain of the $i$ -th decision variable.             |
| $F$                           | Fitness function.                                    |
| $s$                           | A solution vector.                                   |
| $s_i$                         | Value of the $i$ -th variable in $s$ .               |
| $X, S, P$                     | A collection of solutions                            |
| $x^\psi, y^\psi, s^\psi$      | Partial Solutions (PS) / patterns.                   |
| $*$                           | Wildcard value (unfixed variable).                   |
| $s_i^\psi$                    | Value of the $i$ -th variable in $s^\psi$ .          |
| $P_{\text{Ref}}$              | Reference Population.                                |
| $\text{match}(x^\psi, s)$     | Boolean function matching $x^\psi$ to $s$ .          |
| $P_{\text{match}}$            | Subset of $P_{\text{Ref}}$ matched by $x^\psi$ .     |
| $P_{\text{notmatch}}$         | Subset of $P_{\text{Ref}}$ not matched by $x^\psi$ . |
| $\text{apply}(x^\psi, s)$     | Applies fixed variables of $x^\psi$ to $s$ .         |
| $\text{fixed\_count}(x^\psi)$ | Number of fixed variables in $x^\psi$ .              |

## CONTENTS

---

|           |                                                      |    |
|-----------|------------------------------------------------------|----|
| <b>I</b>  | <b>Introduction</b>                                  |    |
| 1         | Introduction                                         | 2  |
| 1.1       | Definition of Partial Solutions and Linkage Learning | 2  |
| 1.2       | Hypothesis                                           | 3  |
| 1.3       | Brief Context on BT                                  | 3  |
| 1.4       | Brief summary of work                                | 3  |
| 1.5       | Considerations and Challenges                        | 4  |
| 1.5.1     | Absence of an explicit model                         | 4  |
| 1.5.2     | Absence of a baseline                                | 4  |
| 1.5.3     | Speed                                                | 4  |
| 1.5.4     | Relevance                                            | 4  |
| 1.6       | Research Directions                                  | 4  |
| 1.6.1     | Research Questions                                   | 4  |
| 1.6.2     | Research Objectives                                  | 5  |
| 1.6.3     | Contribution                                         | 6  |
| 1.7       | Structure of thesis                                  | 6  |
| <b>II</b> | <b>Background</b>                                    |    |
| 2         | Background on Optimisation and Metaheuristics        | 10 |
| 2.1       | Optimisation Problems                                | 10 |
| 2.1.1     | Definition                                           | 10 |
| 2.1.2     | Terminology                                          | 10 |
| 2.1.3     | Notation                                             | 11 |
| 2.1.4     | Types of optimisation problems                       | 11 |
| 2.1.5     | Challenges of solving optimisation problems          | 13 |
| 2.2       | Benchmark Problems used in this research             | 15 |
| 2.2.1     | OneMax                                               | 16 |
| 2.2.2     | Unitary problems                                     | 16 |
| 2.2.3     | Graph Colouring                                      | 17 |
| 2.2.4     | RoyalRoadWithOverlaps                                | 18 |
| 2.2.5     | Max-SAT problems                                     | 19 |
| 2.2.6     | NK Problems                                          | 19 |
| 2.2.7     | The Staff Rostering problems                         | 19 |
| 2.3       | Metaheuristics                                       | 22 |
| 2.3.1     | General introduction to metaheuristics               | 22 |
| 2.3.2     | Real World applications of metaheuristics            | 22 |
| 2.3.3     | Terminology and classifications                      | 23 |
| 2.3.4     | Notation and Definitions                             | 24 |
| 2.3.5     | General Framework                                    | 24 |
| 2.3.6     | Operators                                            | 25 |
| 2.3.7     | Single Objective methods                             | 30 |
| 2.3.8     | Estimation of Distribution Algorithms                | 32 |
| 2.3.9     | Multi-Objective methods                              | 33 |
| 2.3.10    | Source of mistrust                                   | 35 |
| 2.3.11    | Schema theory                                        | 36 |
| 3         | Background on Explainability                         | 38 |
| 3.1       | History of explanations                              | 39 |
| 3.2       | Essential terminology                                | 39 |
| 3.3       | Motivations                                          | 40 |
| 3.3.1     | Motivations specific to BT                           | 43 |
| 3.4       | Definitions and characterisation of explanations     | 44 |
| 3.4.1     | A priori and A posteriori                            | 44 |
| 3.4.2     | Explanandum and Explanans                            | 44 |
| 3.4.3     | Format of explanations                               | 45 |
| 3.4.4     | Local and Global explanations                        | 45 |
| 3.4.5     | Model-agnostic and model-specific                    | 45 |
| 3.5       | State of the art of Explanation methods              | 45 |
| 3.5.1     | Transparent models                                   | 46 |
| 3.5.2     | Variable importance methods                          | 48 |

|        |                                                                                      |    |
|--------|--------------------------------------------------------------------------------------|----|
| 3.5.3  | Linkage Learning methods . . . . .                                                   | 49 |
| 3.6    | Explanations and metaheuristics . . . . .                                            | 49 |
| 3.6.1  | XAI within metaheuristics . . . . .                                                  | 50 |
| 3.6.2  | Metaheuristics for XAI . . . . .                                                     | 50 |
| 3.7    | Challenges and issues within the field . . . . .                                     | 51 |
| 3.7.1  | Are explanations useful? . . . . .                                                   | 51 |
| 3.7.2  | Plausible but not accurate, is that deceptive? . . . . .                             | 51 |
| 3.7.3  | Transparent is not necessarily understandable . . . . .                              | 52 |
| 3.7.4  | Illusion of Understanding . . . . .                                                  | 52 |
| 3.8    | Evaluation of explainability . . . . .                                               | 52 |
| 3.8.1  | What makes evaluation so challenging? . . . . .                                      | 53 |
| 3.8.2  | Evaluation for methods similar to metaheuristics . . . . .                           | 53 |
| 4      | State of the Art . . . . .                                                           | 54 |
| 4.1    | Schemata . . . . .                                                                   | 54 |
| 4.2    | Backbones . . . . .                                                                  | 54 |
| 4.3    | Innovization by Deb . . . . .                                                        | 55 |
| 4.4    | Search Trajectory Networks and Local Optima Networks . . . . .                       | 55 |
| 4.5    | Human in the Loop Approaches . . . . .                                               | 55 |
| 4.6    | Surrogate Model Mining . . . . .                                                     | 56 |
| 4.7    | Anchors . . . . .                                                                    | 56 |
| 4.8    | Learning Classifier Systems . . . . .                                                | 57 |
| 4.9    | Variable importance methods . . . . .                                                | 57 |
| 4.10   | Search Trajectory Mining . . . . .                                                   | 57 |
| 4.11   | Contributions and novelty in context . . . . .                                       | 58 |
| III    | Methodology: Partial Solutions . . . . .                                             |    |
| 5      | Introduction to Partial Solutions . . . . .                                          | 62 |
| 5.1    | Brief introduction . . . . .                                                         | 62 |
| 5.2    | Summary of Papers . . . . .                                                          | 63 |
| 5.3    | Origin . . . . .                                                                     | 63 |
| 5.4    | Formal Definition . . . . .                                                          | 63 |
| 5.4.1  | Properties of Partial Solutions . . . . .                                            | 63 |
| 5.5    | Search spaces for Partial Solutions . . . . .                                        | 64 |
| 5.6    | Post-processing of Partial Solutions . . . . .                                       | 64 |
| 5.7    | A practical example . . . . .                                                        | 65 |
| 5.7.1  | The problem . . . . .                                                                | 65 |
| 5.7.2  | Solving the problem and generating the dataset . . . . .                             | 66 |
| 5.7.3  | Methodology: loading the dataset . . . . .                                           | 66 |
| 5.7.4  | Finding Partial Solutions . . . . .                                                  | 68 |
| 6      | Mining of Explanations as Partial Solutions . . . . .                                | 73 |
| 6.1    | Introduction . . . . .                                                               | 73 |
| 6.2    | Summary of methodology . . . . .                                                     | 73 |
| 6.3    | Motivation behind the overarching methodology . . . . .                              | 74 |
| 6.4    | Acquisition of the Reference Population . . . . .                                    | 75 |
| 6.5    | Search operators for Partial Solutions . . . . .                                     | 76 |
| 6.5.1  | Preliminaries: Operators used in the early stages of research . . . . .              | 76 |
| 6.5.2  | Operators for Any-Pattern (global) search of Partial Solutions . . . . .             | 77 |
| 6.5.3  | Operators on Subset-of-Solution (local explanations) for Partial Solutions . . . . . | 79 |
| 6.6    | Multi-Objective Problem and Search Method . . . . .                                  | 82 |
| 6.6.1  | Experiments to determine the most suitable Multi-objective method . . . . .          | 82 |
| 6.6.2  | Budget and Other Parameters . . . . .                                                | 84 |
| 6.6.3  | Resulting Pareto Front . . . . .                                                     | 84 |
| 6.7    | Search metrics for Partial Solutions: Fitness quality metrics . . . . .              | 86 |
| 6.8    | Simplicity metrics . . . . .                                                         | 91 |
| 6.9    | Search metrics for Partial Solutions: Atomicity metrics . . . . .                    | 93 |
| 6.9.1  | Average bivariate atomicity . . . . .                                                | 93 |
| 6.9.2  | Motivation behind atomicity . . . . .                                                | 93 |
| 6.9.3  | Atomicity + Independence . . . . .                                                   | 94 |
| 6.10   | Other metrics . . . . .                                                              | 95 |
| 6.11   | Illustrative Examples on Benchmark Problems . . . . .                                | 95 |
| 6.11.1 | Motivating mean-fitness, simplicity and atomicity . . . . .                          | 96 |
| 6.11.2 | Knowing when to mine Partial Solutions: OneMax . . . . .                             | 97 |

|        |                                                                                       |     |
|--------|---------------------------------------------------------------------------------------|-----|
| 6.11.3 | Issues with PS mining: Trapk . . . . .                                                | 98  |
| 6.12   | Post-processing via descriptors . . . . .                                             | 99  |
| 6.12.1 | Descriptors example . . . . .                                                         | 100 |
| 6.12.2 | Formalising the concept of “relatively high / low” . . . . .                          | 100 |
| 6.12.3 | Mathematical definition . . . . .                                                     | 101 |
| 6.12.4 | Design considerations behind descriptors . . . . .                                    | 102 |
| 6.12.5 | Expected usage for descriptors . . . . .                                              | 102 |
| 6.13   | Clustering of patterns . . . . .                                                      | 103 |
| 6.14   | Execution time . . . . .                                                              | 103 |
| 6.15   | Effect of Reference population on Mining process . . . . .                            | 106 |
| 7      | Linkage Learning . . . . .                                                            | 108 |
| 7.1    | Introduction . . . . .                                                                | 108 |
| 7.1.1  | The metaheuristic context . . . . .                                                   | 109 |
| 7.1.2  | Pairwise linkage for computational speed . . . . .                                    | 109 |
| 7.2    | Characterisations of Linkage . . . . .                                                | 109 |
| 7.2.1  | Fitness-informed vs fitness-independent . . . . .                                     | 109 |
| 7.2.2  | Paired and unpaired samples approaches . . . . .                                      | 110 |
| 7.2.3  | <b>Symmetric or Not Symmetric</b> . . . . .                                           | 112 |
| 7.2.4  | Whether it is <b>probability of linkage</b> , or <b>strength of linkage</b> . . . . . | 112 |
| 7.2.5  | Differing definitions of non-linkage . . . . .                                        | 112 |
| 7.3    | Overview of Linkage Learning Methods . . . . .                                        | 112 |
| 7.3.1  | Non-linearity Detection Methods . . . . .                                             | 113 |
| 7.3.2  | Fitness independent methods . . . . .                                                 | 114 |
| 7.3.3  | Other Methods . . . . .                                                               | 114 |
| 7.4    | Novel Linkage methods . . . . .                                                       | 114 |
| 7.4.1  | Mutual Information based on survival probability . . . . .                            | 114 |
| 7.4.2  | Dominance based Linkage . . . . .                                                     | 115 |
| 7.5    | Evaluation of linkage learning . . . . .                                              | 120 |
| 7.5.1  | Challenges in evaluation, and their resolution . . . . .                              | 120 |
| 7.5.2  | Evaluating linkage detection: automatic threshold selection . . . . .                 | 121 |
| 7.5.3  | Benchmark problems used . . . . .                                                     | 122 |
| 7.5.4  | Metaheuristics used . . . . .                                                         | 124 |
| 7.5.5  | Linkage methods being evaluated . . . . .                                             | 125 |
| 7.5.6  | Experimental results . . . . .                                                        | 125 |
| 7.5.7  | Conclusion . . . . .                                                                  | 129 |
| 8      | Applications of Partial Solutions . . . . .                                           | 131 |
| 8.1    | Explaining metaheuristic solutions: use case on Staff Rostering . . . . .             | 131 |
| 8.1.1  | Summary of methodology . . . . .                                                      | 131 |
| 8.1.2  | Methodology . . . . .                                                                 | 131 |
| 8.1.3  | Local Explanations . . . . .                                                          | 132 |
| 8.1.4  | Utility of local explanations . . . . .                                               | 133 |
| 8.1.5  | Global Explanations . . . . .                                                         | 133 |
| 8.1.6  | Utility of global explanations . . . . .                                              | 134 |
| 8.1.7  | Evaluation of metaheuristic explanations . . . . .                                    | 134 |
| 8.2    | Explanations for recommender models . . . . .                                         | 135 |
| 8.2.1  | Recommender model . . . . .                                                           | 136 |
| 8.2.2  | Resolution of challenges . . . . .                                                    | 136 |
| 8.2.3  | Evaluation . . . . .                                                                  | 141 |
| 8.2.4  | PS4XRS Conclusions . . . . .                                                          | 142 |
| 8.3    | Partial Solutions used in prediction models . . . . .                                 | 144 |
| 8.3.1  | What is the issue with existing transparent surrogate models? . . . . .               | 144 |
| 8.3.2  | Methodology . . . . .                                                                 | 145 |
| 8.3.3  | Experiments . . . . .                                                                 | 146 |
| 8.3.4  | Results . . . . .                                                                     | 147 |
| 8.4    | How the Partial Solutions relate to Explainability . . . . .                          | 153 |
| 8.4.1  | What is being explained? . . . . .                                                    | 153 |
| 8.4.2  | Are the patterns understandable? . . . . .                                            | 153 |
| 8.4.3  | Are the explanations true? . . . . .                                                  | 153 |
| 8.4.4  | Do Partial Solutions have practical advantages? . . . . .                             | 154 |
| 8.4.5  | Is the process interactive and customisable? Is there trial and error? . . . . .      | 154 |
| 8.4.6  | Is there a ground truth? Is it known if this system finds the ground truth? . . . . . | 154 |

|        |                                                                                                    |     |
|--------|----------------------------------------------------------------------------------------------------|-----|
| 9      | User Study                                                                                         | 155 |
| 9.1    | Motivations                                                                                        | 155 |
| 9.2    | Our specific scenario                                                                              | 155 |
| 9.2.1  | Resource constraints                                                                               | 156 |
| 9.3    | Design process                                                                                     | 157 |
| 9.3.1  | Use case scenarios                                                                                 | 157 |
| 9.3.2  | Planning of study metrics and audience                                                             | 157 |
| 9.3.3  | Design of Experimental procedure                                                                   | 157 |
| 9.3.4  | Simplification of the original optimisation problem                                                | 158 |
| 9.3.5  | Solving the problem and generation of the user materials                                           | 158 |
| 9.3.6  | Design of questions                                                                                | 158 |
| 9.3.7  | Generation of explanations                                                                         | 159 |
| 9.3.8  | Implementation of experiment                                                                       | 160 |
| 9.3.9  | Deployment of user study                                                                           | 160 |
| 9.4    | Outcome of study and analysis of results                                                           | 160 |
| 9.4.1  | Analysis of subjective questions                                                                   | 160 |
| 9.4.2  | (Hypothetical) Quantitative analysis of Task 1                                                     | 164 |
| 9.4.3  | (Hypothetical) Quantitative analysis of Task 2                                                     | 164 |
| 9.5    | On the inconclusive outcome of the study                                                           | 164 |
| 9.6    | Conclusion                                                                                         | 164 |
| 10     | Testimonial from British Telecom Researcher                                                        | 165 |
| 10.1   | Interviewee Background and Context                                                                 | 166 |
| 10.1.1 | Black-box metaheuristics are necessary                                                             | 166 |
| 10.1.2 | Black-box metaheuristics are not trusted                                                           | 166 |
| 10.2   | Task without explanations                                                                          | 166 |
| 10.2.1 | The interface                                                                                      | 169 |
| 10.2.2 | The role of fitness                                                                                | 169 |
| 10.2.3 | Task completion complexity                                                                         | 169 |
| 10.2.4 | Trust and human explanations for manual modifications                                              | 169 |
| 10.3   | Task with explanations (Linkage Learning and Partial Solutions)                                    | 170 |
| 10.3.1 | Task strategy with explanations                                                                    | 170 |
| 10.3.2 | Interface user experience                                                                          | 173 |
| 10.3.3 | Advantage of explanations                                                                          | 173 |
| 10.3.4 | Trust and human explanations for manual modifications                                              | 174 |
| 10.4   | Missing elements                                                                                   | 174 |
| 10.4.1 | Conclusions from Testimonial                                                                       | 174 |
| IV     | Conclusion                                                                                         |     |
| 11     | Discussion, Future work and Conclusion                                                             | 176 |
| 11.1   | Answers to the research questions                                                                  | 176 |
| 11.2   | Discussion of advantages and disadvantages                                                         | 177 |
| 11.3   | Future Work                                                                                        | 178 |
| 11.3.1 | Continuing the user study                                                                          | 178 |
| 11.3.2 | Expanding applicability to continuous and other kinds of problems                                  | 178 |
| 11.3.3 | Expanding applicability to machine learning                                                        | 179 |
| 11.3.4 | Partial Solutions for search performance: a PS-based Estimation of Distribution Algorithm          | 179 |
| 11.3.5 | A Human-in-the-loop optimiser based on PSs                                                         | 180 |
| 11.3.6 | A PS-based Learning Classifier System                                                              | 180 |
| 11.4   | Final remarks                                                                                      | 180 |
| V      | Appendices                                                                                         |     |
| A      | Appendix A: Extended version of the “Mining Potentially Explanatory Patterns via Partial Solutions | 202 |
| B      | Appendix B: Algorithms for various metrics                                                         | 212 |
| B.1    | On the representation of Partial Solutions                                                         | 212 |
| B.2    | Finding solutions which match a pattern, and their fitness                                         | 212 |
| B.3    | Algorithm to find solutions which differ by a single variable                                      | 213 |
| B.3.1  | Naïve Approach                                                                                     | 213 |
| B.3.2  | Optimized Approach using Hash Collisions                                                           | 214 |
| C      | Appendix C: Definitions of Linkage                                                                 | 216 |
| C.1    | Notation for linkage                                                                               | 216 |
| C.2    | Linkage Learning methods                                                                           | 216 |

|       |                                                                        |     |
|-------|------------------------------------------------------------------------|-----|
| c.2.1 | The F-link example function . . . . .                                  | 216 |
| c.3   | Issues with non-linearity detection methods and their issues . . . . . | 217 |
| c.3.1 | Inability to quantify linkage . . . . .                                | 217 |
| c.3.2 | The effect of non-linearity . . . . .                                  | 217 |
| c.4   | Perturbation-based linkage . . . . .                                   | 218 |
| c.4.1 | Perturbation linkage definition . . . . .                              | 218 |
| c.4.2 | Walsh decomposition . . . . .                                          | 220 |
| c.5   | Fitting linear models and ANOVA . . . . .                              | 220 |
| c.5.1 | Building a naive linear model with interactions . . . . .              | 220 |
| c.5.2 | Building a more sensible linear model . . . . .                        | 220 |
| c.5.3 | ANOVA linkage definition . . . . .                                     | 221 |
| c.6   | SHAP interaction via XGBoost . . . . .                                 | 222 |
| c.6.1 | SHAP linkage definition . . . . .                                      | 222 |
| c.7   | Correlation . . . . .                                                  | 223 |
| c.8   | $\chi^2$ tests . . . . .                                               | 224 |
| c.8.1 | $\chi^2$ linkage . . . . .                                             | 224 |
| c.9   | Mutual Information and Bayesian linkage . . . . .                      | 224 |
| c.10  | Friedman's H statistic . . . . .                                       | 225 |
| c.11  | Cosine similarity linkage . . . . .                                    | 226 |
| D     | Appendix D: Linkage tables . . . . .                                   | 228 |
| E     | Appendix E: Benchmark Staff Rostering problems . . . . .               | 237 |
| E.1   | Definition of SR-RR . . . . .                                          | 237 |
| E.2   | Definition of SR-GC . . . . .                                          | 237 |
| F     | Appendix F: Questionnaire for User Study . . . . .                     | 239 |
| G     | Appendix G: Ethics approval . . . . .                                  | 244 |

## LIST OF FIGURES

|            |                                                                                                                                                                                                                                                                                                                                                     |     |
|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| Figure 1.1 | Summary of chapter structure in this thesis, with clickable links.                                                                                                                                                                                                                                                                                  | 6   |
| Figure 2.1 | In non-convex problems, local optima are not always global optima. . . . .                                                                                                                                                                                                                                                                          | 12  |
| Figure 2.2 | A Pareto front for a multi-objective problem of two functions to be minimised. Pareto Optimal solutions such as A and B dominate solutions outside of the Pareto front (such as C), and they are non-dominating between each other. . . . .                                                                                                         | 13  |
| Figure 2.3 | Example of an optimisation problem. This problem needs at least 3 colours to have no-same colour connections, and the configuration shown is optimal. . . . .                                                                                                                                                                                       | 18  |
| Figure 2.4 | The rota patterns used for the simplified Staff Rostering (SR) problem. Specifically, there were the rotas shown to group B.                                                                                                                                                                                                                        | 21  |
| Figure 2.5 | The workers in an instance of the simplified SR problem. Workers which are connected (or in the same enclosed area) share a skill. . . . .                                                                                                                                                                                                          | 21  |
| Figure 2.6 | A flow chart of a typical population-based metaheuristic. It is an iterative process, where selection, crossover and mutation are used to generate new solutions. . . . .                                                                                                                                                                           | 24  |
| Figure 3.1 | This figure represents explainability as many plasters attempting to resolve the issues caused by black box methods. These many aims often overlap. . . . .                                                                                                                                                                                         | 38  |
| Figure 3.2 | Summary of the taxonomy provided in [12], only including the sections which are relevant to this thesis. . . . .                                                                                                                                                                                                                                    | 46  |
| Figure 3.3 | Example of a decision tree, as shown in [225] . . . . .                                                                                                                                                                                                                                                                                             | 47  |
| Figure 4.1 | An example of an anchor, provided in [228]. A model was trained to determine whether a person would earn more or less than \$50'000 a year, and the rule shown guarantees a certain prediction. . . . .                                                                                                                                             | 56  |
| Figure 5.1 | Given a population of solution, we find Partial Solution (PS) that can describe it well . . . . .                                                                                                                                                                                                                                                   | 62  |
| Figure 5.2 | Plot showing the detected linkage strength through “Domination-based” linkage learning. The cell at [row]-[column] has a high value (lighter colour) if the item <row> has an effect on fitness dependent on <column>. . . . .                                                                                                                      | 68  |
| Figure 6.1 | Distribution for a proxy-fitness function that is discrete, and the corresponding distribution for its average (over the space of Partial Solutions). . . . .                                                                                                                                                                                       | 101 |
| Figure 7.1 | The two different main kinds of linkage learning. In “fitness-informed” linkage, each variable affects fitness, but the effect of one variable is dependent on another. In “fitness-independent” linkage detection, it is the values that are affected: the value of a variable is (at least partially) determined by the value of another. . . . . | 110 |
| Figure 7.2 | Evaluation strategy for linkage detection. Each yellow rectangle is a table with ground truth linkage information and measured linkage values. . . . .                                                                                                                                                                                              | 123 |
| Figure 7.3 | Graph Colouring problem instance used for evaluation. The problem contains variables that are in differently sized cliques, and even variables that are not connected to anything. . . .                                                                                                                                                            | 123 |
| Figure 7.4 | Ground truth linkage for benchmark problems . . . . .                                                                                                                                                                                                                                                                                               | 124 |
| Figure 7.5 | Plot showing how the runtime of each linkage method grows as the number of variables increases. The size of the dataset was kept constant at 20000. . . . .                                                                                                                                                                                         | 129 |
| Figure 7.6 | Tables showing the average run time in seconds for each linkage learning method, varying the quantity of variables and the size of the $P_{Ref}$ used. Darker colours represent longer times. Crosses represent situations where the algorithm took longer than 30 seconds. . . . .                                                                 | 130 |

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |     |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| Figure 8.1  | Three explanations produced by the system. The patterns are presented in the blue regions, whereas the descriptors are presented in the yellow regions. The percentile of each descriptor is also presented. . . . .                                                                                                                                                                                                                                                                                                                                                                                                   | 133 |
| Figure 8.2  | A scatter-plot to show the correlation between the frequency of workers, each corresponding to a variable, appearing in <b>PSs</b> , and the quantity of rota choices which are available to them. A light horizontal jitter was applied. . . . .                                                                                                                                                                                                                                                                                                                                                                      | 133 |
| Figure 8.3  | Example of explanation produced for PS4XRS. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 135 |
| Figure 8.4  | Summary of methodology used for Partial Solutions For Recommender Systems ( <b>PS4XRS</b> ). The initial dataset is reduced in dimensionality using clustering, and this is then treated as a $P_{Ref}$ . After mining <b>PSs</b> from this dataset, they are post-processed into explanations. . . . .                                                                                                                                                                                                                                                                                                                | 137 |
| Figure 8.5  | An example of a decision tree that is produced by our methodology, of depth 3. Here, a solution is passed to the model by checking which patterns it contains (which are the explanations), and a leaf node suggests the predicted fitness. . . . .                                                                                                                                                                                                                                                                                                                                                                    | 144 |
| Figure 8.6  | A comparison of how the search method affects the decision tree model. These scatterplots compare the true fitness of solutions against the predictions, along with a correlation line. Our decision trees strongly correlate with the ground truth when the search method is a Genetic Algorithm ( <b>GA</b> ), Simulated Annealing ( <b>SA</b> ). The data shown is for <b>GC_anna</b> , which was solved using 4 different search methods, and the evaluated solutions were split into training and test data (80%-20%). . . . .                                                                                    | 148 |
| Figure 8.7  | Box-plots showing the distribution of the “complexity” of the splitting criteria at different depths for PS-SW (blue) and PS-SWA (orange, on the right) for the benchmark problems (where depth 0 is the root node). These are for instances where the problem was solved using a GA, with 100 trials per combination. . . . .                                                                                                                                                                                                                                                                                         | 149 |
| Figure 9.1  | An example of an explanation provided to the user. The explanation consists of a shallow decision tree based on Partial Solutions (right). The tree structure might also be interpreted as a narrowing of the range of possible fitnesses of the solution (left), so both formats are presented. . . . .                                                                                                                                                                                                                                                                                                               | 156 |
| Figure 9.2  | The “tips and tricks” section of the material available to the user, used to provide them a mental model of how they might tackle the optimisation problem . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                   | 159 |
| Figure 9.3  | Histograms of questionnaire responses. Key insights indicate that participants are confident in their technical skills and generally accept automated methods for administrative tasks (e.g., work rotas), but remain skeptical of their use in critical applications. . . . .                                                                                                                                                                                                                                                                                                                                         | 161 |
| Figure 10.1 | Best solution viewer tools, available to our industrial collaborator during the interview. The fitness function is broken into its summands (range scores for every combination of skill and weekday), and these are shown as a heatmap (top of figure). These range scores are based on how many workers with each skill appear throughout the 3 month period, as shown in the mid-section of the figure. The rota assignments for each worker, along with other rota patterns available and skillset, are shown in the bottom-left of the figure. The user was able to compare rota patterns (bottom-right). . . . . | 167 |
| Figure 10.2 | Solution modification tool. This allowed the user to apply small modifications to the best known solution, and view the effect on fitness. The user could select multiple workers and change each of their rotas. The effect (fitness change) of each individual change is shown on the left, and the overall change in fitness (subject to linkage) is shown on the right. . . . .                                                                                                                                                                                                                                    | 168 |
| Figure 10.3 | Last-minute emergency tool. This produced a random emergency, where an employee was not able to work on the assigned rota pattern. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 168 |

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |     |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| Figure 10.4 | Linkage learning viewer. The user was able to decide which linkage learning method to use (Perturbation by default), and view the resulting linkages as a heatmap (middle section of figure). The user was also able to select individual workers, and see which other workers they were linked to (bottom section of figure). . . . .                                                                                                                                                                                                                                                                                      | 171 |
| Figure 10.5 | PS mining settings selection interface. This allowed the user to perform a PS mining search with the desired fitness evaluation budget, population size, search space (local or global explanations), objectives and constraints). The user could enforce (via the constraints) that the pattern needed to assign certain workers a given rota, or to avoid a given rota. . . . .                                                                                                                                                                                                                                           | 172 |
| Figure 10.6 | Partial Solution viewer. This tool allowed to view the PSs found by the PS mining process. The PSs are ordered according to the criterion selected by the user (minmax regret by default, other options were the objectives used for the PS search). The user could also optionally cluster the items shown. Each PS could be selected to show its pattern, and descriptors could be generated by clicking on “Calculate descriptors”. The settings used for calculating descriptors were selectable by the user (top of figure, by default the descriptor threshold was 0.1, and the quantity of samples was 1000. . . . . | 172 |
| Figure C.1  | A Trap-k problem (3 cliques of 5 bits) was solved by a standard GA, and the resulting set of solutions is analysed for linkage using ANOVA. On the left, the standard fitness function is used, whereas on the right the fitness function is the sigmoid ( $\sigma(x) = 1 + \frac{1}{1+\exp(x)}$ ) of the original fitness ( $F'(x) = \sigma(F(x))$ ). The sets of solutions are perfectly identical, and the only difference is in their fitness. . . . .                                                                                                                                                                  | 218 |
| Figure C.2  | Diagram showing the elements involved in Joint Entropy, Conditional Entropy and Mutual Information. Mathematical notation is intentionally avoided in order for the reader to focus on what elements are used in each formula. . . . .                                                                                                                                                                                                                                                                                                                                                                                      | 225 |
| Figure C.3  | Example of a Bayesian Network for an optimisation problem with 6 variables. Each variable is a node, and the arrows indicate conditional distributions. For example, $V_2$ ’s value distribution is conditional to $V_1$ ’s value. $V_3$ is conditional to the combination of values of $V_2$ and $V_6$ . The distribution of $V_5$ is independent of other variables. . . . .                                                                                                                                                                                                                                              | 226 |
| Figure D.1  | Linkage tables representing the pairwise linkage relationships between the variables of each benchmark optimisation problem. Yellow cells mark pairs of variables which are linked. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                 | 228 |
| Figure E.1  | Rota choices available for . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | 237 |
| Figure E.2  | Graph Colouring problem corresponding to SR-GC . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | 238 |
| Figure E.3  | Workers and their rotas corresponding to the first clique in the Graph Colouring problem. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | 238 |
| Figure G.1  | Proof of ethics approval . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 244 |

## LIST OF TABLES

|           |                                                                                                                                                                                                                                                                                                       |     |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| Table 2.1 | Notation used in optimisation . . . . .                                                                                                                                                                                                                                                               | 11  |
| Table 2.2 | Notation used in optimisation . . . . .                                                                                                                                                                                                                                                               | 16  |
| Table 5.1 | Table showing the name, calories, cost, weight and volume of each purchasable item in the example optimisation problem. The volume refers to the percentage of space which an item would occupy in a bag-for-life. . . . .                                                                            | 67  |
| Table 6.1 | Table showing the results of the comparison between different Multi-Objective evaluation methods. The percentage shows how often the procedure was able to find all of the expected <b>PSs</b> . Figures in bold show the best performing Multi-objective ( <b>MO</b> ) method for every row. . . . . | 83  |
| Table 6.2 | Example Reference Population table, with proxy-fitness column                                                                                                                                                                                                                                         | 100 |
| Table 6.3 | Example Reference Population table, with descriptor column                                                                                                                                                                                                                                            | 100 |
| Table 6.4 | Average runtime ( $\pm$ standard deviation) for each combination of $N$ and <b>PS</b> mining objectives tested, when the <b>PS</b> evaluation budget = 10000. . . . .                                                                                                                                 | 104 |
| Table 6.5 | Average runtime ( $\pm$ standard deviation) for Objectives = SMA, across combinations of $N$ , $K$ , and <b>Pref_size</b> . . . . .                                                                                                                                                                   | 105 |
| Table 6.6 | Average runtime ( $\pm$ standard deviation) for Objectives = SM, across combinations of $N$ , $K$ , and <b>Pref_size</b> . . . . .                                                                                                                                                                    | 105 |
| Table 6.7 | Average runtime ( $\pm$ standard deviation) for Objectives = $M(A+I)$ , across combinations of $N$ , $K$ , and <b>Pref_size</b> . . . . .                                                                                                                                                             | 106 |
| Table 6.8 | Average proportion of discovered groups for the SR-RR and SR-GC problems, applied on differing reference populations (Best performing for each column is shown in <b>BOLD</b> ) . . . . .                                                                                                             | 106 |
| Table 7.1 | Example of a scenario where an optimisation problem with three variables ( $u$ , $v$ and $w$ ) and a fitness function $F$ is investigated. All the solutions in the search space are shown, but the linkage learning method only has access to a sub-sample of them (rightmost column). . . . .       | 111 |
| Table 7.2 | Outcome partition table where $a$ and $b$ are independent, in a RoyalRoadProblem . . . . .                                                                                                                                                                                                            | 117 |
| Table 7.3 | Outcome partition table where $a$ and $b$ are linked, in a Royal-Road problem . . . . .                                                                                                                                                                                                               | 118 |
| Table 7.4 | Overview of linkage learning methods. . . . .                                                                                                                                                                                                                                                         | 119 |
| Table 7.5 | Example of a table to be used for evaluation of linkage. Every row corresponds to a pair of variables, for which we know if they are linked or not (True = linked, False = non-linked), along with the linkage strength that was measured. . . . .                                                    | 121 |
| Table 7.6 | The best balanced accuracy for each linkage detection, when using data from every benchmark problem and every meta-heuristic. The entries are sorted from best to worst . . . . .                                                                                                                     | 126 |
| Table 7.7 | The best balanced accuracy is presented for every combination of linkage detection method and every benchmark problem. In bold is the highest number of each column (i.e. for each problem). . . . .                                                                                                  | 126 |
| Table 7.8 | Table showing the best balanced accuracy for each linkage detection method on each metaheuristic. Figures in bold are the highest for their column. . . . .                                                                                                                                           | 127 |
| Table 7.9 | Best threshold (yielding the best BBA) for each linkage detection method. The “Threshold” column indicates the threshold that yields the best balanced accuracy. These thresholds are those which yield the best balanced accuracy shown in Table 7.6. . . . .                                        | 128 |
| Table 8.1 | Data sparsity for different quantities of clusters. For every cluster of products, there is a certain probability that a given user has interacted with it. . . . .                                                                                                                                   | 137 |

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |     |
|-----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| Table 8.2 | Table showing how much “information” each dataset retains when using certain clustering quantities and model evaluation methods. A model (either a linear model (LR) or an Multi-Layer Perceptron (MLP)) was trained and evaluated on the dataset, and the $R^2$ value is reported. These rows are ordered by $R^2$ descending. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 138 |
| Table 8.3 | Statistics for different ways of obtaining the PSs for $N = 250$ . Here, we test different objectives, match thresholds and search operators. In the objectives, S = Simplicity (pattern size), U = User Count, M = Mean Fitness, A = Atomicity. In threshold, $t = \text{auto}$ means that $t$ was evolved, and $t = \text{off}$ means that the entire pattern had to match. In the search operators, TS = Traditional Sampling, TM = Traditional Mutation, TC = Traditional Crossover. RS, RM and RC are the customised counterparts. The metrics reported are quantity of patterns per run, pattern size, proportion of sessions matched (coverage), Hit Rate (HR). Means and SD are reported, since having a good spread is in some cases more important than the average itself. . . . . | 143 |
| Table 8.4 | (average) $R^2$ score for each decision tree, applied Staff Rostering, GC_anna and GC_jean. Values in bold are the highest per row (i.e. which decision tree performed best). . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 150 |
| Table 8.5 | (average) $R^2$ score for each decision tree, applied to SAT_100, SAT_50 and SAT_20. Values in bold are the highest per row (i.e. which decision tree performed best). . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 151 |
| Table 8.6 | The performance of PS-SW trees and traditional trees is compared. For every combination (problem, depth, metaheuristic), we report the difference in average $R^2$ values and the p-value for the test of differences of averages (Mann-Whitney-U, single-tailed). Cells in bold are those which have a p-value below 0.005. . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 152 |
| Table 9.1 | Qualitative Codebook for Task 1 Textual Explanations . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 162 |
| Table 9.2 | Qualitative codebook for participants’ long-form answers in Task 2 (without explanations). . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | 163 |

## LIST OF ACRONYMS

---

|               |                                                 |
|---------------|-------------------------------------------------|
| <b>AI</b>     | Artificial Intelligence                         |
| <b>ANOVA</b>  | Analysis of Variance                            |
| <b>BMDA</b>   | Bivariate Marginal Distribution Algorithm       |
| <b>BOA</b>    | Bayesian Optimisation Algorithm                 |
| <b>BT</b>     | British Telecom                                 |
| <b>DARPA</b>  | Defence Advanced Research Projects Agency       |
| <b>DEUM</b>   | Distribution Evaluation Using Markov network    |
| <b>DSMGA</b>  | Dependency Structure Matrix Genetic Algorithm   |
| <b>EC</b>     | Evolutionary Computation                        |
| <b>ECDF</b>   | Empirical Cumulative Distribution Function      |
| <b>EcGA</b>   | Extended Compact Genetic Algorithm              |
| <b>EDA</b>    | Estimation of Distribution Algorithm            |
| <b>FOS</b>    | Family of Subsets                               |
| <b>GA</b>     | Genetic Algorithm                               |
| <b>GOMEA</b>  | Gene pool Optimal Mixing Evolutionary Algorithm |
| <b>GP</b>     | Genetic Programming                             |
| <b>HITL</b>   | Human-In-The-Loop                               |
| <b>IEC</b>    | Interactive Evolutionary Computation            |
| <b>IML</b>    | Interpretable Machine Learning                  |
| <b>LCS</b>    | Learning Classifier System                      |
| <b>ML</b>     | Machine Learning                                |
| <b>MLP</b>    | Multi-Layer Perceptron                          |
| <b>MO</b>     | Multi-objective                                 |
| <b>NSGA</b>   | Non-Dominated Sorting Genetic Algorithm         |
| <b>PCA</b>    | Principal Component Analysis                    |
| <b>PS</b>     | Partial Solution                                |
| <b>PS4XRS</b> | Partial Solutions For Recommender Systems       |
| <b>RSPS</b>   | Recommender System Partial Solutions            |
| <b>SA</b>     | Simulated Annealing                             |
| <b>SR</b>     | Staff Rostering                                 |
| <b>SVM</b>    | Support Vector Machine                          |
| <b>XAI</b>    | Explainable Artificial Intelligence             |



Part I

INTRODUCTION

**Optimisation problems are unavoidable:** maximisation and minimisation tasks arise naturally in many domains, including many critical applications [230]. These problems are highly diverse, which has led to a great variety in how they are solved. Some exact methods exist when a task can be modelled mathematically in its entirety, but in many cases the only feasible method is **metaheuristics**.

Metaheuristics is a term used to describe solvers for optimisation problems which are at their core trial and error: random guesses are made and evaluated, and this information informs the next guesses to be made. Because randomness is so central to these methods, they are characterised by stochastic behaviour: the results vary between runs. Metaheuristics were invented in the 1950s, and are highly capable of solving a wide variety of optimisation problems [133]. Although the implementation of these methods can be small and well understood, the emerging complexity arising from the optimisation process results in a quantity of information which cannot be feasibly understood by a human user, often making these methods **black boxes**.

This severely limits the ability to understand, verify, trust, or adapt the outputs. Because metaheuristics are increasingly considered for critical applications, those ethical and practical issues are preventing further adoption of metaheuristic methods [300].

A concrete example of this issue comes from our industrial partner, British Telecom (BT), which needs to allocate its engineers to working rotas. It is necessary to maximise the resulting “evenness” of the workforce allocation, which makes the task an optimisation problem. Because of its scale and complexity, BT has developed a metaheuristic to solve it, but its application results in ethical and practical issues due to its uninterpretability.

This thesis aims to resolve the aforementioned issues by developing novel explainability methods for metaheuristics.

## 1.1 DEFINITION OF PARTIAL SOLUTIONS AND LINKAGE LEARNING

The concepts of **Partial Solution** and **Linkage Learning** are necessary in order to fully understand the first 4 chapters of this document. Only an intuition is required for these initial chapters, after which we provide more in-depth definitions in Chapter 5.

A **Partial Solution** is a pattern in the decision variables of combinatorial solutions. The solutions [2 3 1 4], [1 3 1 1] and [0 3 1 4] all contain the pattern [\* 3 1 \*]. The solution [0 0 0 0] does not contain that pattern. Partial Solutions are conditions where every decision variable is either ignored (\*) or *must* be a given value. The pattern [\* 3 1 \*] is the condition “the solution’s second variable must be a 3, AND the third must be a 1”.

A PARTIAL SOLUTION is just a pattern and nothing else. It can be used to refer to groups of solutions, and that is its main purpose. Most PSs are not useful or noteworthy, but some PSs represent a group of solutions which are desirable (i.e. they have high fitness values). Finding these “noteworthy” patterns is a complex task, which will be discussed throughout this thesis.

Additionally, PSs can be easier to “process” than solutions, because they have the same format but omit some (and potentially most) variables. In the pattern [\* 3 1 \*] we only care about 2 variables out of 4. This aspect becomes more important as the quantity of decision variables increases: we may be able to find “small patterns” even in problems with hundreds of decision variables.

PARTIAL SOLUTIONS CAN BE USED FOR EXPLANATIONS because they allow us to make statements about groups of solutions using a simple rule. The solution to a complex problem with 150 decision variables may be impossible to understand, but our system could find that solutions presenting the pattern [\* 6 3 \* \* \* ...] have fitness values in the top 10% of all solutions. This pattern describes a group of solutions we are interested in.

Although the fitness function and the overall problem are humanly intractable, a human may be able to understand how setting the second and the third variables to 6 and 3 are the main drivers behind the optimality of solutions. It is a **useful approximation** of the fitness function, which serves as an explanation.

LINKAGE LEARNING helps in finding desirable **PSs**. Linkage learning is the process of finding **which decision variables interact with each other** in influencing fitness. In the example given above, the pattern `[* 6 3 * * * ...]` involves 2 decision variables. Perhaps the second variable being set to 6 has a negative effect in general, but when the third variable is set to 3 it has a highly positive effect on fitness values. Because finding useful **PSs** is hard, linkage learning is used as a guide to which patterns are most likely to be useful. For example, if two variables are not linked, we should not look into patterns which set both of their values.

## 1.2 HYPOTHESIS

The hypothesis being investigated by this thesis is as follows:

We hypothesise that solutions produced by metaheuristics for combinatorial optimisation problems can be made more understandable. This is achieved by breaking them into smaller patterns, which we call **Partial Solutions**. The fitness of a solution can be understood as the combined effects of the Partial Solutions present.

Regarding Partial Solutions, a more specific hypothesis is investigated:

We hypothesise that it is possible to find Partial Solutions associated with high fitness which are also understandable. These Partial Solutions can then be used for explanations. Partial Solutions can also be used to implement transparent prediction models, but also to analyse datasets such as those produced by a Recommender System.

## 1.3 BRIEF CONTEXT ON BT

Our industrial partner British Telecom (**BT**) encounters black-box optimisation issues when solving their Staff Rostering problem. They have reported several specific issues:

- Metaheuristic results often need to be modified manually to fit unforeseen constraints. These results are hard to modify without significantly degrading their quality.
- **BT** faces resistance in adopting metaheuristics due to a lack of trust in their results from Human Resources and regulatory bodies.
- Users often ignore metaheuristic suggestions due to a lack of trust, and instead hand-craft the solutions. This produces worse results overall.
- There is an inability to understand how problem settings (workers available, rotas available) could be modified to improve solutions.

The Staff Rostering (**SR**) problem is a combinatorial optimisation task, featuring highly-dimensional solutions (141 variables in the instance analysed in [46]) and a complex fitness function. The problem is solved using a customised algorithm (iRoster, [81]).

These aspects make the solutions hard or impossible to understand, which motivates the need for explanations. We chose not to develop an approach specialised on the **SR** problem or iRoster, and instead we developed a problem-agnostic and metaheuristic-agnostic method to expand its applicability. The **BT** context served as an initial set of research questions, and the methods developed were then developed to be generally applicable.

## 1.4 BRIEF SUMMARY OF WORK

This thesis implements **explainability** features for optimisation problems, achieved using Partial Solutions (**PSs**). These are patterns that describe groups of solutions, and we aim to find **PSs** associated with high-fitness solutions. Simultaneously, these patterns must be simple and easy to understand. For a solution such as `[2 3 1 6]`, a Partial Solution could be `[2 * * 6]`. The “less important” parts of the solutions have been removed in order to make the solutions easier to understand. For an optimisation problem with combinatorial solutions belonging to a certain search space, Partial Solutions are

defined as vectors of values from the same search space, but every vector can now also have the value “\*”.

The **PSs** of interest are those which are Pareto optimal with respect to multiple objectives, which make them suitable for explanations. Such objectives can include complexity, effect on fitness, and linkage strength. Once these desirable **PSs** are found, they are post-processed into explanations.

**PSs** are a medium for producing explanations, and their flexibility allows them to adapt to a variety of situations. They are used as standalone explanations for metaheuristics, but they were also applied to a dataset derived from a Recommendation System model. **PSs** were also used to create transparent models which predict the fitness of solutions.

In order to evaluate the utility of our explanations, a user study was designed to statistically prove whether these provide an advantage.

## 1.5 CONSIDERATIONS AND CHALLENGES

This section outlines the main considerations and challenges encountered during this research. This context is necessary for understanding the research questions.

### 1.5.1 *Absence of an explicit model*

Many explainability methods within Explainable Artificial Intelligence (**XAI**) were developed for Machine Learning (**ML**). These methods assume a model is present; however metaheuristics usually lack such a **model**. Here, “model” refers to higher-level information learned from input data, and explaining **ML** involves representing this learned information usefully.

Although some metaheuristics do possess such a model (Estimation of Distribution Algorithms (**EDAs**), discussed in Section 2.3.8), our methodology must be applicable to model-free approaches like Simulated Annealing (**SA**) and a wide range of other search methods.

### 1.5.2 *Absence of a baseline*

Although some explanations could be considered “correct” for certain benchmark problems, there is no universal criterion for determining what makes an explanation satisfactory. An explanation’s quality is dependent on its audience, and even then the question of “what makes the explanation good” is a subjective matter.

### 1.5.3 *Speed*

**BT** indicated that their workflow requires explanations in under two minutes. This is not a hard constraint, but it has serious implications: faster execution significantly reduces the barrier to usability. An explainability method which is too slow will not be used as much, and thus its benefits are voided.

### 1.5.4 *Relevance*

Explanations for optimisation problems and metaheuristics aim to answer different (although similar) questions to those for **ML**. Therefore, the **XAI** techniques already developed for **ML** will often be inapplicable for our aims.

## 1.6 RESEARCH DIRECTIONS

This section formalises the research questions guiding this thesis and outlines the concrete objectives required to answer them.

### 1.6.1 *Research Questions*

This thesis investigates methods to explain metaheuristic solutions by using **PSs**. The focus is on explaining the *solutions* rather than the metaheuristic itself. This addresses

a significant research gap, while also mitigating the practical concerns associated with black-box search methods. We formulate the following primary research questions:

- **RQ1: How can the results of a metaheuristic be explained using PSs?** Specifically, how can explanations facilitate various tasks relating to using the solutions?
- **RQ2: How can PSs be defined** for explanations, and how can they be generated efficiently from a set of solutions to maximise their explanatory power?
- **RQ3: How effective are PSs in supporting decision making?**

These primary questions focus on explainability. However, additional secondary questions are also of interest:

- **RQ4: Can PSs act as models to predict fitness**, potentially in an explainable manner?
- **RQ5: Can PSs be applied to other contexts beyond optimisation ?**

### 1.6.2 Research Objectives

The research questions will be answered by pursuing the following specific research objectives:

- **O1:** Formally define PSs and the metrics used to quantify their explanatory suitability.
  - PSs should occupy a niche not covered by existing explainability methods. As discussed in Section 3.7, variable importance dominates XAI but assumes linearity. Therefore, PSs are specifically designed for problems with complex variable interactions.
  - Define PSs and their metrics formally to ensure the proposed methods are easily replicable.
  - The explanations must avoid common XAI issues, such as a lack of provability or demonstrable utility [94].
  - The PSs should harness **linkage learning** in order to expose the underlying structure of the optimisation problem. Linkage learning itself requires further research in order to be satisfactory for our purposes, in terms of quality of results and execution speed.
- **O2:** Develop algorithms to generate Partial Solution-based explanations quickly.
  - Each explanation should be generated in the order of seconds, rather than minutes or hours.
  - Scalability is a major concern. The SR problem for example features large solutions and is highly non-linear (see Section 2.2.7.1).
  - Ensure PSs are flexible: problem-agnostic, metaheuristic-agnostic, and adaptable to other domains. We only assume that the problem is combinatorial. Further assumptions on the problem will prevent general applicability.
  - The metaheuristic must be treated as a black box. We cannot assume the presence of a population (for example BT’s iRoster is a single-individual SA) or of an explicit model. The only guarantee is that the solver evaluates many solutions. Similarly, there are no assumptions about the distribution of the solution being evaluated, which allows our methodology to be decoupled from the metaheuristic. In domains like Recommender Systems where a metaheuristic is missing entirely, our methodology should still be applicable.
- **O3:** Evaluate the explanatory power of the system.
  - The evaluation methods must align with XAI literature.
  - The evaluation should prove practical utility in use cases relevant to BT.
  - The evaluation should acknowledge the possibility of researcher bias and take steps to prevent it.

- **O4:** Create Partial Solution-based models which are both transparent and effective for predictions.
  - The resulting models should avoid the common misconception that an understandable format leads to understandable models. The size of the model should be small in order to be feasibly understandable.
- **O5:** Find generalisations for PSs and our general methodology, in order to align with the desiderata of stakeholders. Our methodology should be easily modifiable whilst remaining anchored to its central procedure.

### 1.6.3 Contribution

The research objectives culminate in the following core contributions:

- **C1:** A pattern-based definition of PSs, detailed in Section 5.4. These patterns are optimised against the metrics defined in Section 6.8 for understandability, Section 6.7 for fitness quality, and Section 6.9 for “structural integrity”. These objectives are themselves a significant contribution.
- **C2:** A multi-objective search method with customised operators to obtain PSs, as discussed in Chapter 6
- **C3:** The integration of domain-specific information to improve PS interpretability, discussed in Section 6.12.
- **C4:** A novel, transparent method for predicting solution fitness using PS-based decision trees, shown in Section 8.3.
- **C5:** A methodology for evaluating metaheuristic explanations designed to mitigate researcher bias, in the form of a user study described in Chapter 9.
- **C6:** Two novel linkage learning methods, which are designed to work on data produced by metaheuristics, defined in Section 7.4.
- **C7:** An evaluation of many linkage learning methods, defined in Section 7.5.

## 1.7 STRUCTURE OF THESIS

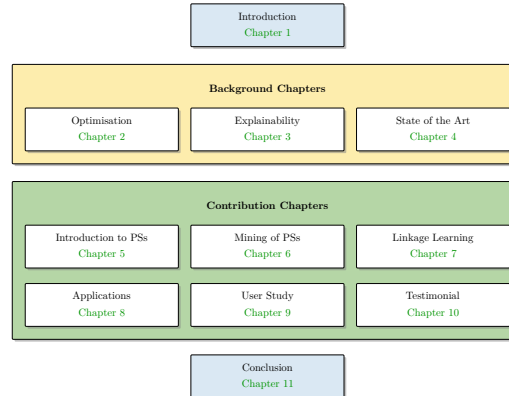


Figure 1.1: Summary of chapter structure in this thesis, with clickable links.

The structure of the thesis is shown in Figure 1.1, and is summarised as follows:

- **(Chapter 1) Introduction** : Introduces the motivation, problem context, and research questions of the thesis.
- Background chapters:
  - **(Chapter 2) Background on optimisation** : Covers the fundamental concepts of combinatorial optimisation and the search methods used to solve them.
  - **(Chapter 3) Background on explainability** : describes how explanations have been researched in the context of computer science, and provides definitions and concepts used throughout the thesis.
  - **(Chapter 4) State of the Art**: Compares existing methods against our proposed methodology, and discusses its novelty.
- Contribution chapters:
  - **(Chapter 5) Introduction to Partial Solutions** : introduces the concept of PSs and provides formal definitions. It also includes a worked example of their usage.
  - **(Chapter 6) Mining of Partial Solutions**: documents the methodology developed to find useful PSs. It describes and motivates the multi-objective search method, its operators, and its objectives.
  - **(Chapter 7) Linkage Learning**: A review of existing Linkage Learning methods, followed by proposed novel approaches and their evaluation.
  - **(Chapter 8) Applications of Partial Solutions**: contains the main contribution, all based on PSs. They include post-hoc explanations for optimisation problems (e.g. Staff Rostering), transparent prediction models, and explanations for Recommender Systems. This chapter also evaluates each method.
  - **(Chapter 9) User Study**: the evaluation of our explanations through a user study.
  - **(Chapter 10) Testimonial**: our explanation system was shown to our industrial collaborator, and a report of his testimonial is given
- **(Chapter 11) Conclusions**: closing remarks, discussion of how the research questions have been answered, and future work.
- Appendices:
  - (Chapter A) Original paper defining PS Mining.
  - (Chapter B) Algorithms used to implement various PS objectives.
  - (Chapter C) Linkage learning methods definitions.
  - (Chapter D) Example linkage tables
  - (Chapter E) Benchmark SR problems used for evaluation of PS mining.
  - (Chapter F) User study questionnaire described in Chapter 9.
  - (Chapter G) Ethics approval letter.



## Part II

# BACKGROUND

Optimisation problems are at the core of this thesis, since they are the subject of the explanations. These problems are characterised by emergent complexity: small definitions can give rise to complex and diverse classes of optimisation problems.

In order to discuss the different varieties of optimisation problems, Section 2.1 will formally define them in mathematical terms.

The hardest part is solving these problems, and for this thesis we are interested in how **metaheuristic methods** are used for this purpose. This topic is covered in detail in Section 2.3.

The reason why optimisation problems and metaheuristics are being researched here is that their results are often **unexplainable**, meaning that it is not clear how they were obtained or why they work. A short discussion on this is presented in Section 2.1.5.5, and a comprehensive background on the topic of **explainability** is given in Chapter 3.

## 2.1 OPTIMISATION PROBLEMS

Optimisation problems occur naturally in real-world applications, across many fields. In chemistry, one may find the optimal reaction conditions, find drugs that satisfy many objectives, or find the optimal lab experiment scheduling. In engineering, one can find the optimal design of vehicles and circuits. In computer science, one can find the optimal decision variables for a machine learning model [9].

### 2.1.1 Definition

Optimisation problems are tasks described by a **collection of decision variables** and an **objective function**. For a given combination of decision variables (a “**solution**”), the objective function describes its quality (called the “objective” or “**fitness**”). The aim is to find the combination(s) of decision variables that result in the best fitness value, which is the minimum for a minimisation problem, and the maximum for a maximisation problem. More formally, a maximisation task is one where the aim is to find  $x$  such that:

$$x = \arg \max_{x \in X} F(x) \quad (2.1)$$

$F$  is the objective, and  $X$  is the set of all possible solutions, the **search space**. If the intention is to minimise the objective, a similar equation is used:

$$x = \arg \min_{x \in X} F(x) \quad (2.2)$$

To note is that these formulations are equivalent: multiplying the objective of a maximisation problem by  $-1$  and finding the minimising  $x$  has the same results:

$$\arg \max_{x \in X} F(x) = \arg \min_{x \in X} -F(x) \quad (2.3)$$

A more formal formulation of optimisation problems can be found in [28]. The rest of this section will focus on single-objective optimisation problems, and other variants are covered throughout the rest of the chapter.

For the rest of the thesis, objectives are to be **maximised**.

### 2.1.2 Terminology

Here we clarify the terminology used throughout the thesis.

**optimisation problem/task** : aiming to find the input which maximises a certain quality metric.

**objective / fitness / utility metric** : measures the quality of solutions.

**solution / candidate solution** : a possible input for the objective function. In this thesis, a solution will always be a fixed-length vector.

**decision variables** : the numbers that together form a solution.

**values** : the possible values which can be taken by a decision variable in a solution

**cardinality** : of a decision variable, it is the quantity of possible values it can take.

**search space** : the set of all possible solutions, but more practically it is a description of the decision variables and the values they can take.

**fitness landscape** : the values taken by the fitness function over the entire search space

**fitness assessment / evaluation** : computing of the objective value of a solution

**search** : the process of looking for the optimal solution. This is at times called **solving** the problem.

**global optimum** : the best solution, because it maximises or minimises the objective

**local optimum** : a solution that is better than all of its “neighbours”, which are solutions similar to it.

### 2.1.3 Notation

Table 2.1 shows the symbols used to indicate the items described above.

| Table 2.1: Notation used in optimisation |                                                     |
|------------------------------------------|-----------------------------------------------------|
| Symbol                                   | Meaning                                             |
| $F$                                      | Fitness function                                    |
| $x, y, z, s, t$                          | Individual solutions                                |
| $L$                                      | Length of solutions in the search space             |
| $x = (x_1, x_2, \dots, x_L)$             | A solution with $L$ decision variables              |
| $i, j, a, b, k$                          | Indices of decision variables                       |
| $\text{cardinality}(i)$                  | Number of possible values for decision variable $i$ |
| $X, S, P$                                | Collections of solutions                            |

### 2.1.4 Types of optimisation problems

Optimisation problems can be categorised by the **type of decision variables**, the **shape of the fitness landscape** and the **quantity of objectives**.

#### 2.1.4.1 By type of decision variables

A common categorisation in the literature of optimisation consists of **combinatorial** and **continuous** problems [220].

A problem is **combinatorial** if the decision variables in the solution can take values from a finite set, such as {Red, Green, Blue}, {True, False} and {1, 2, 3}. These fall under the category of **discrete problems**, because the size of the search space is finite.

If the decision variables can take real values from a range, such as [1, 4] (which includes 1.1, 2, 3.9999), then the problem is **continuous**. The size of the search space for these is theoretically infinite, whereas for combinatorial problems it is very large but still finite [153]. There are many other kinds of problems, where the solutions are variable-length lists, trees, text, permutations etc..., but these are not relevant to the present work.

The main problem which motivated this research, the Staff Rostering (SR) problem described in Section 2.2.7, is a combinatorial problem.

#### 2.1.4.2 Convex vs non-convex

This classification relates to the fitness landscape, and has strong implications for the complexity of the problem. As mentioned in Section 2.1.2, there are **local** and **global optima**, and in a **convex problem** these are always the same, as shown in Figure 2.1. Informally, these are problems where a solution can be proven to be the global optimum by showing that any slight modification will worsen its fitness [32].

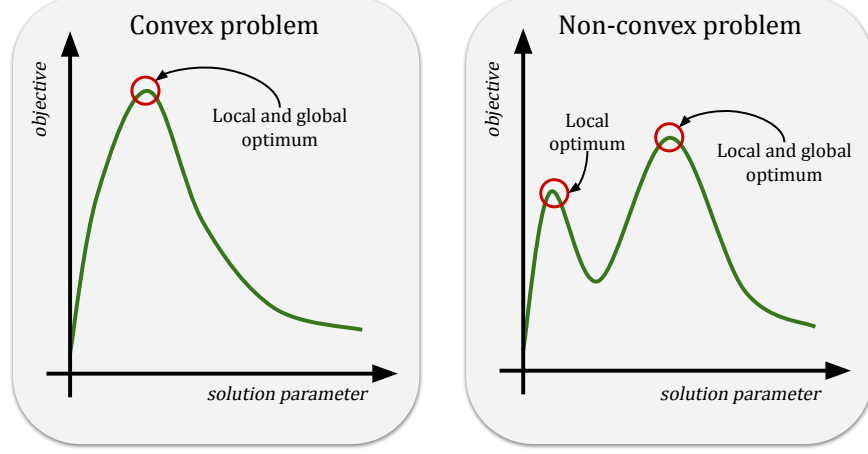


Figure 2.1: In non-convex problems, local optima are not always global optima.

In **non-convex problems**, the same proofs are not valid: although the neighbourhood of a solution might only have worse fitness, by going far enough it might be possible to discover solutions with better fitness, as shown in Figure 2.1.

This difference makes it much harder to determine when a global optimum is found in a non-convex problem, and thus this category contains the most computationally complex problems which are solved via **global optimisation methods**. Generally, convex problems follow polynomial complexity, whereas many non-convex problems are NP-hard. [68]. Non-convex problems are one of the main driving factors behind the development of metaheuristics, as purely mathematical methods are often inapplicable.

More detailed and formal definitions of convex and non-convex problems are found in [32, 68].

#### 2.1.4.3 By Quantity of Objectives

The problems discussed so far were **single-objective** problems, where there is a single objective to be optimised. While these are more typical, there also exist **multi-objective** problems, in which multiple and often conflicting objectives must be considered simultaneously [185].

In single-objective optimisation, solutions can typically be ordered from “best to worst” according to their fitness (mathematically, they follow a **total order**). A solution is either better, worse or equivalent to another in terms of fitness. However, in multi-objective problems, such ordering is generally not possible because improving one objective may come at the expense of another [73].

Here, the term **dominance** is more appropriate: a solution **dominates** another if it is better in all objectives (with some caveats when ties occur, see Equation (2.4)). A solution is **non-dominated** when there are no solutions which dominate it [73]. In notation, dominance is often indicated using  $\succ$ .

$$s \succ t \iff \forall i \in \{1, \dots, m\}, F_i(s) \geq F_i(t) \\ \wedge \exists j \in \{1, \dots, m\}, F_j(s) > F_j(t). \quad (2.4)$$

The set of non-dominated solutions is called the **Pareto front**, and this is what one seeks to find when solving a multi-objective optimisation problem [73]. An example is shown in Figure 2.2. These solutions represent different optimal trade-offs between objectives, and which one is preferable often depends on context-specific priorities or

preferences. It is important to understand that, mathematically, the Pareto front is an inevitable consequence of multi-objective problems with conflicting objectives.

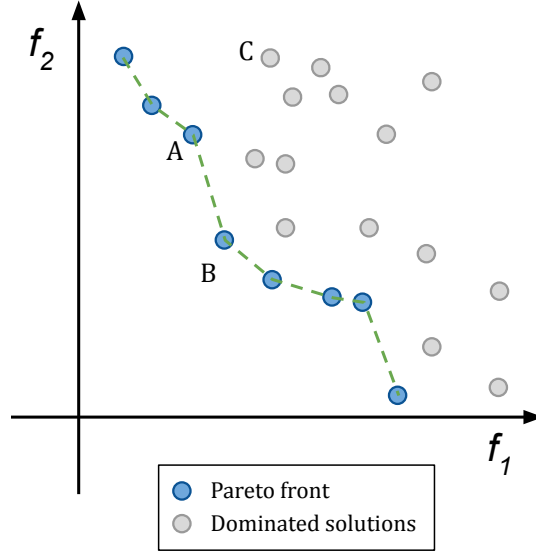


Figure 2.2: A Pareto front for a multi-objective problem of two functions to be minimised. Pareto Optimal solutions such as A and B dominate solutions outside of the Pareto front (such as C), and they are non-dominating between each other.

Therefore, there is rarely one best solution for a multi-objective problem, but this is not a weakness: the Pareto front presents a set of possible options from which a user can decide their most suitable option.

In this thesis, the search for explanations was implemented as the resolution of a multi-objective problem, so this class of problems will be relevant in our methodology.

### 2.1.5 Challenges of solving optimisation problems

There are various challenges to be considered when attempting to solve optimisation problems, and they are summarised here for future convenience.

#### 2.1.5.1 Inability to solve exactly

A naïve method of finding the optimum is **enumeration**: calculating the fitness of the entire search space, and then picking the solution with the best fitness.

Unfortunately, this is in many cases not feasible due to the size of the search space being potentially very large, since it tends to increase exponentially in size as more variables are present (the **curse of dimensionality** [49]). For example, the **SR** problem instance used in our experiments would involve testing  $1.7 \cdot 10^{51}$  candidate solutions.

In some cases, there are exact methods for finding the optimum *without* having to evaluate the entire search space, such as Branch-and-Bound, Dynamic Programming, Integer Programming and Backtracking [137], which do guarantee the optimum. Unfortunately, these methods can still be unfeasible.

Determining if a solution is a global optima (mathematically) is practically impossible for many real-world problems. For many common optimisation tasks, the decision problem of “is this solution the global optimum?” is NP-hard [163], meaning that optimality can only be proven by checking the fitness of every solution in the search space. (Note that this is a simplification, and that although there might be ways to make the process more efficient, it will still be at that unapproachable scale).

Therefore, if a solution with good fitness is found for a problem, knowing whether it is the true optimum might be impossible.

In many cases, finding the global optimum is not strictly necessary as long as a “good enough” solution is found. In these situations, an algorithm which solves the optimisation task would be evaluated based on the fitness of its results rather than its probability of finding the global optima.

In the context of this thesis, the informal term “**good solution**” is used to refer to solutions that have a relatively high fitness in a maximisation problem, and relatively low fitness in a minimisation problem.

#### 2.1.5.2 *Large-scale global optimisation*

A practical issue which is particularly relevant to this thesis is the presence of **large genomes**, meaning that the quantity of variables in a solution is inconveniently big, and the field of **large-scale optimisation** focuses on these [211]. The **SR** problem provided by **BT** as an example of realistic instance consists of solutions with 141 variables, which affects both the search process (curse of dimensionality, as mentioned above) and understandability (too many variables to process or visualise). These 141 decision variables each correspond to engineers working for **BT**.

#### 2.1.5.3 *No free lunch theorem*

The No Free Lunch Theorem is highly relevant when developing methods to solve optimisation problems, and states that [291]:

No optimisation algorithm can outperform any other under any metric over all problems.

The main consequence of this theorem is that if an optimisation method performs better than average for a certain category of problems, it will perform poorly for other categories [151].

For some researchers, a consequence of this theorem is that **optimisation methods cannot be both efficient and transparent**, and thus there is a trade-off between interpretability and efficiency. This has been contested in recent literature, and will be discussed with further detail in Section 3.7.

#### 2.1.5.4 *Assumption of structure*

In order for an optimisation problem to be solvable without enumeration, the objective must exhibit some form of structure. In the case of metaheuristic solvers, this consists in the assumption that similar solutions should have similar fitness, and mathematical methods make much stronger assumptions. Quoting from [149]:

Optimisation algorithms succeed by exploiting structure in the objective function. If no such structure exists, then search reduces to blind guessing.

There are multiple ways in which a problem can display structure, and an efficient algorithm generally attains its performance by specialising on certain structures.

On the other hand, assuming that a problem follows a certain structure can be **detrimental**, which is a consequence of the “No Free Lunch Theorem” [291]. If all problems followed a structure that would allow a more efficient algorithm, then there would be an algorithm that is globally better, which is impossible. This is relevant for this thesis, as we aimed to create a tool which makes few assumptions about the optimisation tasks.

#### 2.1.5.5 *Non-understandability*

Knowing the exact formula for the fitness function does not necessarily mean that a human user will **understand** how to solve it, or that the global optimum will be easy to understand. In many cases it is easier to find the solution to an optimisation problem than to find an *explanation* for its existence.

This issue is one of the main motivations behind this thesis, especially due to real-world optimisation problems being particularly hard to comprehend, as explained in Chapter 3. The sheer quantity of formulas and tables involved prevents a human user from trusting

and using automated methods in solving the optimisation problem. These issues are the main motivations behind the topic of **XAI**, which is discussed in Chapter 3.

There is a fundamental question to be answered: are all optimisation problems in some way understandable?

This question is hard to answer, and too broad for the scope of this thesis. What can make a problem understandable is the possibility of it being approximated, such that it can be understood by a human but at the cost of some lower-importance details. This approximation is a **lossy compression**, and thus compressibility can be taken as an *indication* of an optimisation problem being in some way understandable partially. In this small discussion, we discuss **lossless compressibility**, as this does not require deciding the level of approximation. The two are not the same, but lossless compressibility could be taken as an indication of lossy compressibility, which in turn could be an indication of explanations being possible. These are still just an indication rather than a guarantee, but this discussion is a useful informal starting point in order to argue that most optimisation problems should allow explanations.

First, consider the concept of **Kolmogorov complexity**: every piece of data has a *shortest* description possible. If this shortest description is smaller than the data itself, the data is **Kolmogorov compressible**. Data which cannot be compressed (noise without structure) is **Kolmogorov incompressible** [113]. With optimisation problems, we can say that an optimisation problem’s optimum is explained by showing why it is better than the rest of the search space. If this explanation requires listing the entire search space, then we can consider the problem non-understandable because its fitness function is Kolmogorov incompressible.

On the other hand, we can consider a problem to be understandable if there are more concise explanations (on why the optimum is optimal), due to the fitness function being **Kolmogorov compressible**.

In the real world, we can assume that every fitness function is Kolmogorov compressible, because we know that it must have been implemented using code that is smaller than its search space (barring some unrealistic situations). Therefore, from an Information Theory point of view, all optimisation problems that are encountered in the real world should in some way be understandable (with some loss of lower noise detail). In this sense, **explanations are simply a method of compressing information about solutions**, which is a core idea in this thesis.

#### 2.1.5.6 Interactions and non-separability

Another source of difficulty in optimisation arises from **interaction effects between variables**. In many problems, the impact of modifying one variable depends on the values of others, meaning that variables do not contribute independently to the objective [118]. Knowing which variables are interacting can help in determining how to solve the problem more efficiently, since it could be **separable** into smaller independent problems [265].

The **SR** problem is particularly dense with interactions, due to the workers sharing skills and working days, but finding this information is challenging. Informally, most existing explainability methods are not well defined when interactions are present, and thus this thesis aims to investigate this research gap. Methods for detecting interactions, called **linkage learning**, are discussed in detail in Chapter 7.

## 2.2 BENCHMARK PROBLEMS USED IN THIS RESEARCH

In order to evaluate and compare different resolution methods for optimisation problems, a suite of benchmarks was used throughout this research.

These benchmarks were chosen to cover a wide and diverse array of behaviours that could occur in optimisation problems in the wild. This thesis relates to **combinatorial**

**optimisation problems**, and so all the benchmarks are in the same class, many being **binary**.

Since interaction effects are of interest to this research, almost all the problems display such effects. Generally, there are groups of variables that interact with each other, which we call **cliques** (borrowing from Graph theory terminology, although it has a more specific definition there). The problems in Section 2.2.2 are the main problems used and they are chosen as they are problems displaying interaction effects of different kinds, while also allowing ground truth for what an explanation should look like. Graph Colouring is another one of the main problems used in this work, as it allows for instances which display interaction effects that can mimic what is seen in real-world scenarios, such as what is seen in the **SR** problem. NK problems are used for their “tunable complexity” while Max-SAT problems are used since they are well-known and competing methodologies can be applied and compared against ours.

Note that some of the definitions differ from the literature in how the fitness might be multiplied or divided by a constant. This inconsistency is inevitable, as different papers use different but equivalent definitions. As discussed in Section 2.1, this does not affect the resolution process.

The notation used in this section is the same as that used in Table 2.1, with some additions.  $L$  indicates the quantity of variables,  $F$  is the fitness and  $s$  is a solution. Additionally, we use  $c$  to indicate a clique (as a vector of variables), and `get_clique( $i$ )` to indicate that we are getting a specific clique from the solution. These are summarised in Table 2.2

Table 2.2: Notation used in optimisation

| Symbol                                     | Meaning                                                        |
|--------------------------------------------|----------------------------------------------------------------|
| $F$                                        | Fitness function                                               |
| $L$                                        | Length of solutions in the search space                        |
| $x, y, z, s, t$                            | Individual solutions                                           |
| $x = (x_1, x_2, \dots, x_L)$               | A solution with $L$ decision variables                         |
| $i, j, a, b, k$                            | Indices of decision variables                                  |
| $c$                                        | A clique, which is a collection of variables within solutions. |
| <code>get_clique(<math>i, s</math>)</code> | the $i$ th clique present in the solution $s$                  |
| <code>clique_len</code>                    | The quantity of variables in each clique                       |
| <code>n_cliques</code>                     | The quantity of cliques in each solution                       |

### 2.2.1 *OneMax*

OneMax, defined in [2] is a problem where there are no interactions between variables and each variable has the same contribution. It is a binary problem with  $L$  variables, where the fitness of a solution is just the sum of its values:

$$F(s) = \sum_{i=0}^L s_i \quad (2.5)$$

### 2.2.2 *Unitary problems*

Unitary problems are problems where the variables are grouped into **cliques** [75].

A solution has  $L$  binary variables in total, and its variables are partitioned into `n_cliques` so that each clique consists of `clique_len` variables.

For convenience, the partitioning is so that the first clique is  $(s_0, s_1, \dots, s_{\text{clique\_len}-1})$ , with the second clique starting at  $s_{\text{clique\_len}}$  and so on. We indicate the  $i$ th clique

with `get_clique( $i, s$ )` For each clique, we calculate the sum of its variables, called the **unitation**.

$$\text{unitation}(c) = \sum_{i=0}^{\text{clique\_len}} c_i \quad (2.6)$$

This value is then passed to a sub-function  $F_u$ , and it is this sub-function which distinguishes the various unitary problems.

The final fitness of the solution is then the sum of these sub-fitness values.

$$F(s) = \sum_{i=0}^{\text{n\_cliques}} F_u(\text{unitation}(\text{get\_clique}(i, s))) \quad (2.7)$$

### 2.2.2.1 *RoyalRoad*

RoyalRoad represents the simplest problem where interactions occur. More specifically, a clique has good fitness if and only if all the variables have value 1 [198].

$$F_u^{\text{RR}}(u) = \begin{cases} 1 & \text{if } u = \text{clique\_len} \\ 0 & \text{otherwise} \end{cases} \quad (2.8)$$

### 2.2.2.2 *Trap-k*

Trap-k was designed to be a **deceptive** problem, so that automated solvers would struggle to find the optimum, through these properties:

- There are interaction effects, but only for specific combinations of values
- The general behaviour of the function indicates that more zeros are beneficial, but the best clique is composed entirely of ones.

The unitary function for Trap-k, first discussed in [107], is defined as follows:

$$F_u^{\text{Trap-k}}(u) = \begin{cases} \text{clique\_len} & \text{if } u = \text{clique\_len}, \\ \text{clique\_len} - u - 1 & \text{if } u < \text{clique\_len}. \end{cases} \quad (2.9)$$

Note that this formula replaces the  $k$  used in the literature with `clique_len` to avoid confusion - usually  $k = \text{clique\_len}$ .

More intuitively, in a problem where  $k = 5$ , the optimal clique is [1 1 1 1 1] ( $F_u = 5$ ), closely followed by [0 0 0 0 0] ( $F_u = 4$ ). Adding ones to the latter clique will lower its fitness, but if all the values are set to one then the fitness becomes optimal. Hence, there is no interaction between the variables unless they are all ones.

### 2.2.2.3 *Parity*

This function, called ‘‘Concatenated Parity Function’’ in [59], aims to demonstrate interaction effects which are hard to model for EDAs [59]. The sub-fitness function is:

$$F_u^{\text{Parity}}(u) = u \bmod 2 \quad (2.10)$$

For each clique, half of its possible variable combinations is optimal.

## 2.2.3 *Graph Colouring*

Graph Colouring is a class of problems where there is a network of nodes and edges, and the aim is to colour the nodes so that there are no (or the fewest) connected nodes with the same colour (as seen in Figure 2.3).

This was originally stated as a decision problem, on whether a certain quantity of distinct colours would have allowed for a satisfactory graph colouring, and it has been proven to be NP complete [154].

There are two optimisation tasks stemming from the decision problem:

- Given a graph with nodes to be coloured, what is the fewest quantity of colours so that no two connected nodes have the same colour?
- Given a graph with nodes to be coloured from a limited selection, what is the colour assignment which yields the fewest quantity of connected nodes with the same colour?

The latter of these tasks is widely used to create problems with interaction effects. A summary of theoretical aspects relating to the Graph Colouring problem is given in [146], and a list of real-world instances is given in [270]. An example of a graph colouring problem instance is shown in Figure 2.3.

Compared to other problems presented in this section, this is characterised by non-binary variables, multiple optima, and complex higher-order relationships between variables.

These problems are relevant to this thesis because the behaviour of the **SR** problem often mimics that of a Graph Colouring problem, since workers might have to be assigned to different working patterns so that the workforce is evenly spread. This relationship has long been discussed in the literature, with [253] providing a summary.

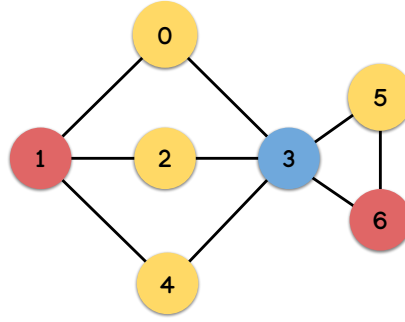


Figure 2.3: Example of an optimisation problem. This problem needs at least 3 colours to have no-same colour connections, and the configuration shown is optimal.

#### 2.2.4 *RoyalRoadWithOverlaps*

This is a novel but minor contribution.

RoyalRoadWithOverlaps was developed as a benchmark for our methodology centred around **PSs**, and was used in [44].

For a RoyalRoad problem instance, the fitness is exactly the quantity of cliques that are entirely made of ones, and thus it can be re-written in a more general form as how many “**optimal\_patterns**” it contains.

$$F(s) = |\{e^\psi \in \text{optimal\_patterns} \text{ s.t. } \text{match}(s, e^\psi)\}| \quad (2.11)$$

(**match**( $a, b^\psi$ ) means that the solution  $a$  contains pattern  $b^\psi$ . This notation will be introduced in Section 6.2)

In RoyalRoad, **optimal\_patterns** might look like  $\{(1 \ 1 \ 1 \ 1 \ * \ * \ * \ *), (* \ * \ * \ * \ 1 \ 1 \ 1 \ 1)\}$ .

This generalisation allows us to use any pattern instead of those that are already present in RoyalRoad. In particular, “overlapping” or “contradicting” patterns can be used. For example, with **optimal\_patterns** =  $\{(* \ 1 \ 1 \ 1 \ 1 \ * \ * \ *), (* \ * \ 0 \ 0 \ 0 \ 0 \ * \ *), (* \ * \ * \ * \ 1 \ 1 \ 1 \ 1)\}$ , the optimum could be  $(1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1)$  but  $(* \ * \ 0 \ 0 \ 0 \ 0 \ * \ *)$  might still a positive trait to have.

This property was used since in the initial stages it was unclear whether it was possible to find patterns which overlap.

### 2.2.5 Max-SAT problems

**Boolean satisfiability problems (SAT)** are defined by a set of Boolean clauses, each expressed in *conjunctive normal form (CNF)*. In the classical decision formulation, the task is to determine whether there exists an assignment of values to the variables such that *all* clauses are satisfied.

In the *optimisation* variant, often referred to as **Max-SAT**, the goal is to find an assignment that satisfies the maximum possible number of clauses. Formally, for a CNF formula  $\varphi = \{C_1, C_2, \dots, C_m\}$  with clauses  $C_j$  the fitness function is:

$$F_{\text{SAT}}(s) = |\{C \in \varphi \mid \text{satisfies}(s, C)\}|$$

SAT was the first class of problems proven to be NP-complete (Cook-Levin theorem, [229, 154]), and Max-SAT inherits the associated complexities.

The instances used for this research are taken from **SATLIB**, a well-established repository of SAT benchmarks inspired by Donald Knuth’s collections of satisfiability instances [135].

### 2.2.6 NK Problems

The *NK* landscape model was originally introduced by Kauffman [155] to capture the structural properties of fitness landscapes in evolutionary biology, particularly relating to proteins and DNA. In modern computer science, it serves as a benchmark for generating tunably rugged combinatorial optimisation problems.

An *NK* problem is a binary problem with  $N$  variables. The landscape’s ruggedness is governed by the **epistatic** parameter  $K$  ( $0 \leq K \leq N - 1$ ), which dictates the degree of interaction among the variables. The core characteristic of the model is that the fitness contribution of an individual variable  $x_i$  is conditionally dependent on its own value as well as the values of  $K$  other variables.

The fitness of a solution is the average of  $N$  localised fitness contributions:

$$F(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N f_i(x_i, x_{i_1}, \dots, x_{i_K}) \quad (2.12)$$

where each component function  $f_i : \{0, 1\}^{K+1} \rightarrow [0, 1]$  assigns a fitness value to the combination of the main variable  $x_i$  and its  $K$  interacting neighbours.

The inputs to  $f_i$  are (generally) sequentially contiguous bits. To handle boundary conditions, the indexing wraps around using modular arithmetic:

$$f_i(\mathbf{x}) = f_i(x_i, x_{(i \pmod{N})+1}, \dots, x_{((i+K-1) \pmod{N})+1}) \quad (2.13)$$

Practically, because the inputs are discrete and binary, each  $f_i$  is implemented as a unique lookup table containing  $2^{K+1}$  randomly generated real numbers, typically drawn from a uniform distribution  $\mathcal{U}(0, 1)$  [217].

### 2.2.7 The Staff Rostering problems

The **SR** problem is the main problem which originated this research. It was first posed by British Telecom (**BT**), and it represents a somewhat simplified version of the real problem faced by managers. There are many versions of this problem, each with different constraints and objectives, and this thesis is concerned with two specific instances:

- A version which is almost identical to that discussed in Fyvie’s work in [100]
- A version which is simplified, used for the user study in [42].

The original “canonical” definition can be found in [81].

#### 2.2.7.1 Original definition

The **SR** optimisation problem consists of assigning rota patterns to a set of workers, where the fitness measures the **consistency** of the quantity of workers over a certain calendar period. The problem is parametrised by:

- A calendar length (e.g. 12 weeks),
- The list of **rota patterns**, each being a boolean vector representing “Working” and “Not Working” days,
- A list of **workers**, each having some rota pattern options and a set of skills (a set of strings). These options are provided by **BT** and are unchangeable. These are picked based on the availability of workers, law compliance and HR acceptance.

Within solutions, each worker is a variable and their rota assignment is their value (e.g. **worker#6**→**option#2**). If **worker#6** has rota options [12, 24, 25, 60], **option#2** will be **pattern#25**.

The fitness function is based on the “range” of the quantity of scheduled employees *with a certain skill on a certain day of the week*. For example, of Fibre Optic engineers on Tuesdays there might be {2,1,5,2,5} throughout five weeks, but having {3,2,2,3,2} would be better since the quantity only ranges between 2 and 3. Let  $Q_x(\text{skill}, \text{weekday})$  be the function returning the quantity of workers trained in **skill** working on **weekday** as determined by solution  $x$ , for each week (e.g.  $Q_x(\text{Fibre}, \text{Tuesday}) = \{2, 1, 5, 2, 5\}$ ).

The aim is to **minimise the range of workers for each skill-weekday combination** (Equation (2.14)), with some additional components that contribute to the objective function in Equation (2.17).

$$\text{Range}_x(\text{skill}, \text{weekday}) = \left( 1 - \frac{\min(Q_x(\text{skill}, \text{weekday}))}{\max(Q_x(\text{skill}, \text{weekday}))} \right)^2 \quad (2.14)$$

$$\text{weight}(\text{weekday}) = \begin{cases} 10 & \text{if } \text{weekday} \in \{\text{Saturday}, \text{Sunday}\} \\ 1 & \text{otherwise} \end{cases} \quad (2.15)$$

$$\text{preference\_score}(x) = W \cdot \#\{\text{non-favourite rotas in } x\} \quad (2.16)$$

$$F(x) = - \left( \sum_{\substack{s \in \text{skills} \\ d \in \text{weekdays}}} \text{Range}_x(s, d) \cdot \text{weight}(d) \right) - \text{preference\_score}(x) \quad (2.17)$$

Firstly, **weekends** have increased importance and thus their contribution to the fitness is increased by using the **weight** function (Equation (2.15)). Additionally, each employee has a **favourite rota** within their choices, and the fitness improves as more of these are picked (Equation (2.16)). Finally, the weighted ranges for each skill-weekday combination are added with the preference score (with a weight  $W$ , here set to 0.001) to form the final fitness function (Equation (2.17)). The final fitness function multiplies all of this by -1 as to make this a **maximisation** problem.

The instance used here is (almost) identical to [97], with the same workers and rotas available to them. There are 141 workers, each having at most 4 rota options of which there are 95 in total. Unlike [97] where **skills** are not considered, there are 18 possible skills and each worker is trained in at most 5. Lastly, the calendar length is 3 months (13 weeks). This problem has interactions, because workers with the same skill have their impact dependent on the allocation of the other workers. This was retained in the simplified version of the problem.

#### 2.2.7.2 User study version

This definition was used for the user study in [42], but it was beyond the scope of the contents of the paper. Therefore, this is its first formal definition.

This problem was developed due to the original **SR** problem being too complex for the expected participants of the user study.

Many things were simplified:

- Every worker has exactly the same rota options
- There are only 4 rota options in total: A, B, C, D (see Figure 2.4),
- The calendar length is 2 weeks,

- The fitness function is much simpler,
- There are only 18 workers, each can have at most 2 skills,
- It can be described and represented (mostly) as a Graph Colouring problem (see Figure 2.5),
- Rotas A and B are designed to be “complementary”, and Rotas C and D are as well.

| Rota option | Rota pattern |     |     |     |     |     |     |        |     |     |     |     |     |     |
|-------------|--------------|-----|-----|-----|-----|-----|-----|--------|-----|-----|-----|-----|-----|-----|
|             | Week 1       |     |     |     |     |     |     | Week 2 |     |     |     |     |     |     |
|             | Mon          | Tue | Wed | Thu | Fri | Sat | Sun | Mon    | Tue | Wed | Thu | Fri | Sat | Sun |
| ROTA A      | ✓            | ✓   | ✓   |     | ✓   | ✓   |     | ✓      | ✓   | ✓   | ✓   |     |     | ✓   |
| ROTA B      | ✓            | ✓   | ✓   | ✓   |     |     | ✓   | ✓      | ✓   | ✓   |     | ✓   | ✓   |     |
| ROTA C      |              | ✓   | ✓   | ✓   | ✓   |     |     | ✓      | ✓   | ✓   | ✓   |     | ✓   | ✓   |
| ROTA D      | ✓            | ✓   | ✓   | ✓   |     | ✓   | ✓   |        | ✓   | ✓   | ✓   | ✓   |     |     |

Figure 2.4: The rota patterns used for the simplified SR problem. Specifically, there were the rotas shown to group B.

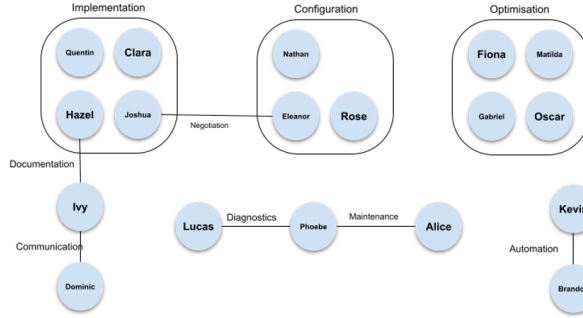


Figure 2.5: The workers in an instance of the simplified SR problem. Workers which are connected (or in the same enclosed area) share a skill.

Formally, the function which counts workers in each weekday-skill combination is almost the same, but this time it also counts which week (first or second week).

$$Q_s(\text{skill}, \text{weekday}, \text{week}) = \text{quantity of workers}$$

with  $\langle \text{skill} \rangle$  on  $\langle \text{weekday} \rangle$  of  $\langle \text{week} \rangle$

Then, the fitness function is the difference in  $Q$  between week 1 and 2.

$$F(s) = \sum_{\substack{\text{skill} \in \text{skills} \\ \text{day} \in \text{weekdays}}} |Q_s(\text{skill}, \text{day}, 1) - Q_s(\text{skill}, \text{day}, 2)|$$

## 2.3 METAHEURISTICS

This thesis will discuss **metaheuristic** methods to solve optimisation problems. These have two roles in our research:

- they are the algorithms to be made more understandable
- they are the method used to generate the explanations

### 2.3.1 *General introduction to metaheuristics*

In simple terms, a metaheuristic is a method of solving optimisation problems characterised by the use of **randomness** as part of the algorithm. In the face of intractably large search spaces, these methods decide to pick certain options at random and are able to find satisfactory solutions. Some of these methods are similar to natural processes, while others are based on statistical models.

The main advantage of these methods is their **flexibility**: as long as a fitness function is implemented these methods can be used, and thus they are the most immediately and widely applicable. This also allows them to be compatible with more exotic tasks, such as generating shapes or even programs. This flexibility does depend on an appropriate representation of solutions being used, but the effort in planning and implementing this is negligible when compared to the effort required by problem-specific algorithms.

On the other hand, they have their issues: they cannot outperform exact methods in quality of results, and they are generally considered to be slower than most mathematical formulations. Moreover, the results produced by a metaheuristic are **stochastic** themselves: running the program with the same settings can lead to different results.

Despite these issues, their flexibility and simplicity are undeniable, making them a valuable tool in computer science.

### 2.3.2 *Real World applications of metaheuristics*

There are countless applications of metaheuristics, thanks to the aforementioned reasons. There are various summaries of their application, such as [230, 28], with a small selection being presented here.

- Medicine [105]
  - Radiotherapy planning, hyperparameter optimisation of clinical models [28]
  - Clinical scheduling of lab personnel [105]
  - Diagnostics of diabetic episodes [105]
- Logistics
  - Vaccine logistics during COVID-19 [222]
  - Vehicle routing, supply chain network optimisation, stocking [281]
  - Beverage logistics [110]
- Biology and chemistry
  - DNA/RNA alignment [103]
  - protein ligand docking and pose prediction (Autodock) [292]
  - Simulation of cancer treatments [223]
- Finance and economics
  - finding optimal investment plans and economic models [56]
  - portfolio management [240]
- Machine learning
  - hyperparameter tuning [6]
  - image analysis [158]
  - software engineering to improve efficiency of tasks [119]
  - reinforcement learning [201]

### 2.3.3 Terminology and classifications

Firstly, a **solver** is a method to find a high-fitness combination of values in an optimisation problem.

The solver will **evaluate** the fitness value of many possible combinations of values. Each combination of values is called a **solution**.

The terms “solving” and “solution” are deceptive, because a “solution” is not necessarily the global optima [261].

Another term commonly used is a **search** method, because there might be a strategy in how these combinations are found.

A **metaheuristic** is a search method which uses stochastic behaviour to find good solutions without making many assumptions about the problem, as opposed to **heuristics** where the procedure makes use of problem-specific properties (such as Dijkstra’s algorithm which can find optimal paths between nodes).

As a term, “*metaheuristics is a rather unfortunate name*” (to quote Sean Luke from [185]), since it would suggest that it is a heuristic to choose heuristics, but they are not. An overview of the different definitions is found in [261], one of which is particularly suitable for this thesis and taken from [27]:

A metaheuristic is a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems. In other words, a metaheuristic can be seen as a general algorithmic framework which can be applied to different optimisation problems with relatively few modifications to make them adapted to a specific problem.

This definition is presented here and used for this thesis because it underlines how metaheuristics are highly diverse and widely applicable but not necessarily monolithic, which are the main properties that were taken into consideration for this thesis.

Metaheuristics harness the power of **randomness**, and due to this they are a major subfield of **stochastic optimisation** [185]. **Stochastic** here means that randomness is used in order to make decisions without a known correct path, which results in algorithms that produce different results even when given the same input, making them **non-deterministic**.

The role played by randomness can be clarified by thinking of metaheuristics as a combination between two strategies: **exploration** and **exploitation** [185].

**Exploration** consists in generating many **diverse solutions**, in order to determine which regions of the search space are more likely to have optimal solutions. This process could be biased to “succeed” more often, but the random sampling aspect is necessary in order to obtain unbiased information.

In **exploitation**, solutions known to have desirable fitness are used as a starting point to obtain even better solutions. Rather than generating solutions randomly, new solutions are slightly modified versions of high fitness solutions. This leans on the fact that *similar* solutions are expected to have *similar* fitness; in these cases the fitness function is termed “well behaved” [261]. By slightly modifying a solution with good fitness it is expected to *eventually* obtain a solution with even better fitness than the original. Due to this, exploitation focuses on **similar** solutions, and it is complementary to exploration.

Randomness plays a role in both aspects: it allows for unbiased sampling of solutions in exploration, and it allows for unbiased modifications in exploitation. Each strategy is commonly associated with respective algorithms [185]:

- **Exploration → Random Search:** Generating uniformly random solutions and evaluating all of them. The solution with the best fitness is returned.
- **Exploitation → Hill Climbing:** Starting from a given solution, many small modifications are applied and we check which of the resulting solutions has improved fitness. These improvement operations are chained until no improvement is available.

These two methods have their complementary faults and merits, and metaheuristics can be thought of as different combinations of the two.

BIO-INSPIRED METHODS **Evolutionary Computation (EC)** is one of the main categories within metaheuristics, where the natural process of evolution is simulated. More specifically, they mimic the natural selection and reproduction of animals within populations [185].

Another term used for these methods is **Genetic Algorithm (GA)**, to underline that there is a data structure which represents a solution (a **representation**) and that this is altered in ways that are similar to DNA mutations [185]. Both EC and GAs belong to the larger category of **bio-inspired** methods, since they borrow ideas from biological phenomena.

#### 2.3.4 Notation and Definitions

In addition to the terms used in optimisation, we introduce terms which are relevant to metaheuristics

**population** : a collection of solutions.

**individual** : another term for solution, especially within a population.

**chromosome / genome / genotype** : the data structure which represents a solution.

**breeding** : the act of producing new solutions by modifying or combining existing ones.

**generation** : an iteration of a population-based method, in which a new population is generated through breeding.

**operator** : a term used to refer to a core operation carried out within a heuristic. Specific operators are discussed in Section 2.3.5.

#### 2.3.5 General Framework

As mentioned before, metaheuristics are a combination of exploration and exploitation. They can be generally described as a combination of certain **operators**.

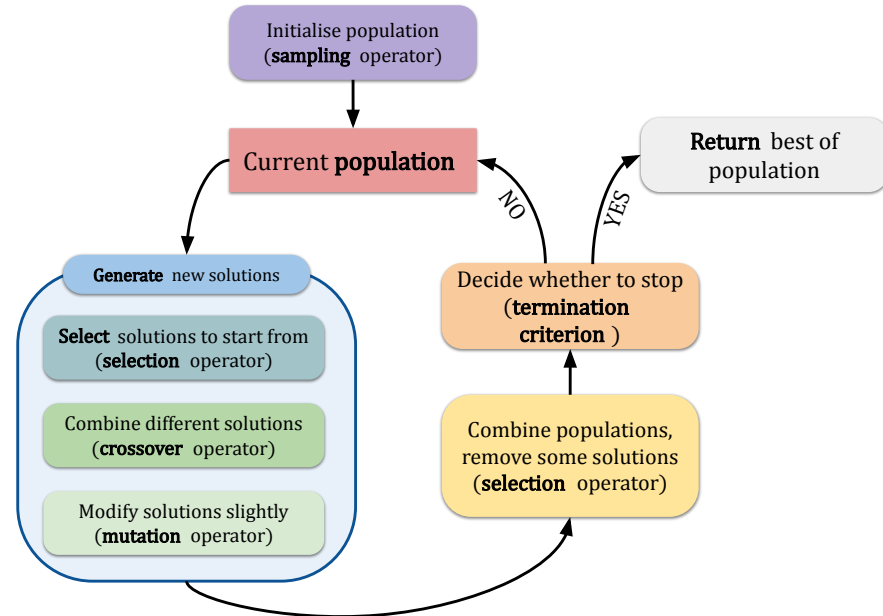


Figure 2.6: A flow chart of a typical population-based metaheuristic. It is an iterative process, where selection, crossover and mutation are used to generate new solutions.

The iterative process implemented by a metaheuristic is composed of:

- **sampling / initialisation operator**: random generation of solutions.
- **breeding operator**: to generate new individuals using previous solutions.
  - In **evolutionary** methods, it is the result of **mutation operator** (small changes to a solution) and **crossover operator** (combining solutions).
  - In **model-based approaches**, it is done by building a model that learns from the objective function, and generating solutions that are expected to have high fitness.
- **perturbation / mutation operator**: a method to create a solution which is slightly different from another solution. The end result is stochastic. This operator is defined in more detail in Section 2.3.6.2.
- **crossover / recombination operator**: a method to combine solutions to produce new ones. This term is derived from chromosomal crossover in nature. Further discussion is provided in Section 2.3.6.3.
- **selection operator**: picking solutions based on their fitness. This is relevant in determining which solutions are the basis for breeding, as well as determining which solutions should be discarded from a population.
- **termination criterion**: what makes the iterative process terminate. This may be based on run time, quantity of generations or count of fitness evaluations.
- **fitness evaluation**: calculating the value of the fitness function for one specific solution.
- **fitness evaluation budget**: the total quantity of fitness evaluations allowed when the termination criterion specifies that the algorithm terminates when a certain quantity of fitness evaluations has occurred.

#### 2.3.5.1 *Population based approaches*

Many metaheuristics are **population-based**, meaning that they store a collection of solutions at the same time rather than just one. This brings many benefits [185]:

- Managing multiple individuals at the same time promotes diversity and allows the implementation of mechanisms which explicitly promote diversity.
- Many individuals being evolved is akin to running a single-individual method multiple times from different starting points, which improves the probability of finding good solutions.
- Having many individuals stored at the same time can allow for “genetic information” to be exchanged between them, similarly to real-world populations. This is done via the crossover operator typically, although the model in an Estimation of Distribution Algorithm (EDA) might also correspond to this.

Population-based methods generally work in **generations**, where in each generation the population is used to construct a replacement population that is hoped to be better. How exactly this process takes place depends on the algorithm, but in some methods the population is changed one individual at a time (**steady state methods**) and thus there are no generations [185].

**Elitism** is a property of some population-based methods, where certain individuals are protected from not being included in the population if they are sufficiently high ranking in terms of fitness [74]. Elitism is used to prevent an iterative metaheuristic from sudden changes in the overall population’s gene pool.

#### 2.3.6 *Operators*

As mentioned before, metaheuristics are characterised by high modularity of **operators**. The content that follows will introduce different kinds of operators, their implementation and when they might be relevant. There is a large quantity of operators represented in the literature, and this section will be focused on those used in combinatorial optimisation.

### 2.3.6.1 Sampling / Initialisation

This is the first operator used by the algorithm, and it is generally quite simple: it generates random initial solutions which will be improved by the algorithm. Most of the time, these are uniformly randomly generated.

In the case of a binary combinatorial problem, each boolean variable can be decided via a virtual coin flip as shown in Algorithm 1, often called **uniform sampling**.

---

**Algorithm 1** Uniform sampling for binary solutions

---

```
1: function UNIFORMBINARYSAMPLING(L)
2:    $s \leftarrow$  array of length  $L$ 
3:   for  $i = 0$  to  $L - 1$  do
4:      $s[i] \leftarrow \text{random\_boolean}()$ 
5:   end for
6:   return  $s$ 
7: end function
```

---

If it is known that a certain area of the search space is likely to contain the optimum, then the sampling operator can take advantage of this information to increase the probability of generated solutions being “near” an optimum.

*[...] if you know something about the likely initial “good” regions of the search space, you could **bias** the random generation to tend to generate individuals in those regions. In fact, you could **seed** the initial population partly with individuals of your own design. [185]*

Therefore, there is an advantage to making use of domain knowledge. At the same time, metaheuristics are often used in order to avoid any bias in case the domain knowledge is not certain. The previous quote continues with:

*[...] Often though you **think** you know where the good areas are, there’s a good chance you don’t. Don’t put all your eggs in one basket: include a significant degree of uniform randomness in your initialisation.*

### 2.3.6.2 Mutation

A mutation operator is used to generate slightly modified versions of existing solutions in a stochastic manner. This mimics the natural phenomenon in which DNA is subject to small changes during asexual reproduction or the inheritance of genes from parents in sexual reproduction.

The expected change between a solution and its mutated counterpart is typically small, and is described as the **mutation rate**. In practical terms, the mutation rate represents the probability of change of each variable in the genome. The exact value of this probability depends on the problem and the operators in use. If the mutation rate used is too low, the metaheuristic is expected to require more mutations (and thus more iterations) to transition between different regions of the search space. If the mutation rate is too high, it might take fewer mutations but also there is an increased chance of “overshooting” desirable regions of the search space [120].

Therefore, there is a trade-off in choosing the value of the mutation rate, although **adaptive mutation rates** can decide this automatically. A literature review on the topic can be found in [120].

In the case of a binary combinatorial problem, the most common operator is **uniform mutation**, where each bit in the solution has a chance of being flipped (that chance being the mutation rate), as seen in Algorithm 2.

Although the ideal mutation value will depend on the situation, the literature generally suggests  $\frac{1}{L}$  to be a suitable choice [185, 15].

### 2.3.6.3 Crossover / Recombination

Crossover exists as a means to mitigate some of the limitations of metaheuristics that rely solely on mutation.

---

**Algorithm 2** Uniform Mutation (binary)

---

```
1: function UNIFORMMUTATION( $s$ , mutation_rate)
2:    $s' \leftarrow$  copy of  $s$ 
3:   for  $i = 1$  to  $L$  do
4:     if random() < mutation_rate then
5:        $s'[i] \leftarrow$  NOT  $s'[i]$ 
6:     end if
7:   end for
8:   return  $s'$ 
9: end function
```

---

Consider a situation where the optimum is composed of multiple complex traits: many variables must take on the correct values before any positive change in fitness can be observed. If solutions evolve “asexually” (only via mutation and selection), the optimum must be discovered within a single “bloodline”, through many individually insignificant mutations that eventually result in the correct trait combination. The probability of these mutations occurring, surviving selection, and being recombined in the same solution is quite low (especially when they are not beneficial to fitness on their own).

This issue is resolved in nature and *in silico* by combining multiple individuals to construct new solutions, which is often termed **sexual reproduction**. Small, individually insignificant mutations can occur in separate individuals and then be combined into a single individual by crossover [185].

A standard crossover operator for binary problems is **uniform crossover**. Two new solutions are generated: initially clones of their parents, but with each variable having a small probability of being swapped with the corresponding variable from the other parent. This probability is called the **crossover rate**.

---

**Algorithm 3** Uniform Crossover (binary)

---

```
1: function UNIFORMCROSSOVER( $a, b$ , crossover_rate)
2:    $c_1 \leftarrow$  copy of  $a$ 
3:    $c_2 \leftarrow$  copy of  $b$ 
4:   for  $i = 1$  to  $L$  do
5:     if random() < crossover_rate then
6:       swap  $c_1[i]$  and  $c_2[i]$ 
7:     end if
8:   end for
9:   return ( $c_1, c_2$ )
10: end function
```

---

Not every new solution needs to be produced via crossover; many metaheuristics use a mixture of sexual and asexual reproduction. This is typically controlled by applying crossover only with a given probability, the **crossover probability** [120].

The terms **crossover rate** and **crossover probability** are often used interchangeably in the literature, but they are different concepts.

Other crossover operators are also popular, such as one-point crossover and two-point crossover, which more closely resemble the natural phenomenon of chromosomal crossover [185]. These exploit the list-based representation of solutions by swapping contiguous sub-segments of the parents.

**LINKAGE AND CROSSOVER** Crossover relates strongly to the concept of **linkage**, where the effect of a variable depends on the values of other variables (also called **epistasis** [286]). Ideally, beneficial combinations of linked variables should be inherited together, but this is not guaranteed by uniform crossover. One-point and two-point crossovers are more effective when linked variables are likely to be adjacent in the solution representation.

While uniform crossover might appear suboptimal due to its tendency to disrupt groups of linked variables, it is also less biased than other crossover operators. One-point and two-point crossover exploit a structural assumption (that linked variables are close in the representation), but this assumption does not necessarily hold for real-world problems.

Every crossover operator is subject to some *representational bias*, but uniform crossover suffers the least [150].

#### 2.3.6.4 Selection

Selection is used to select a subset of a population of solutions, with a bias towards solutions with better fitness. This mirrors the natural effect in which “unfit” individuals are removed from the gene pool by dying or not being able to mate.

How the bias for high quality solutions is implemented can vary greatly, but is generally one of **tournament selection**, **fitness proportionate selection** and **truncation selection**, which are defined later in this section.

These different methods also correspond to different degrees of trade-off between exploration and exploitation [185]. Truncation selection is the more exploitative of the methods, because less fit solutions are immediately discarded, whereas the other two methods are based on probabilities which give a lower but non-zero chance for unfit solutions to remain within the gene pool.

Similarly to biological contexts, the term “selective pressure” in metaheuristics is used to indicate how strongly the fitness of an individual influences their participation to the gene pool [133].

Selection is mainly used in two ways within a bio-inspired metaheuristic:

- finding individuals that will be used as the basis for breeding
- deciding which individuals will be kept in the population and which will be discarded.

How and which selection operators are used depend on the algorithm, and the same algorithm may use multiple. For example, an algorithm may employ tournament selection when deciding the “parents” of a new individual, and then use truncation selection when deciding which individuals are kept in the gene pool.

**TRUNCATION SELECTION** This method is the most exploitative, because the worst solutions in the population are guaranteed to be excluded from selection. This method selects a collection of individuals deterministically: first it sorts the population by fitness, and then selects the  $n$  best [185].

---

#### Algorithm 4 Truncation Selection

---

```

1: function TRUNCATIONSELECTION( $P, n$ )
2:    $P_{\text{sorted}} \leftarrow \text{sort}(P, \text{key} = \text{fitness}, \text{order} = \text{descending})$ 
3:    $P' \leftarrow$  first  $n$  solutions of  $P_{\text{sorted}}$ 
4:   return  $P'$ 
5: end function

```

---

**FITNESS PROPORTIONATE SELECTION** The previous method selected individual solutions with a probability of either 100% or 0%. To introduce more randomness in the system (giving more chances to less-fit individuals), in fitness-proportionate selection the probability of selection is proportional to fitness [132].

There are some technicalities on how this is achieved, since the probability could be based on the fitness or the ranking. In Algorithm 5 the fitness is used directly for simplicity.

**TOURNAMENT SELECTION** The previous approach can be cumbersome, since constructing a list of probabilities and sampling from it with weights is not always efficient. A much simpler approach (producing mathematically similar results) is **tournament selection**.

This method is parametrised by the *tournament size*, which is commonly set to 2 or 3 but without solid justifications [172]. Larger tournaments lead to more exploitative

---

**Algorithm 5** Fitness Proportionate Selection (single solution)

---

```
1: function FITNESSPROPORTIONATESELECTION( $P, f$ )
2:    $f_{\text{norm}} \leftarrow f / \text{sum}(f(P))$ 
3:    $i \leftarrow \text{random.choice}(P, \text{weights} = f_{\text{norm}})$ 
4:    $s \leftarrow P[i]$ 
5:   return  $s$ 
6: end function
```

---

selection pressure, while smaller tournaments increase exploration. Most literature uses small values, but again these are given without empirical evidence; existing literature suggests using self-adaptive parameters instead [172].

---

**Algorithm 6** Tournament Selection (single solution)

---

```
1: function TOURNAMENTSELECTION( $P, k$ )
2:    $C \leftarrow \text{random.sample}(P, k)$ 
3:    $s \leftarrow \text{argmax}(C, \text{key} = \text{fitness})$ 
4:   return  $s$ 
5: end function
```

---

Mathematically, the probability of a solution being selected increases monotonically for higher-ranking solutions in the population, so this method is often mathematically equivalent to fitness-proportionate selection [108].

#### 2.3.6.5 Constraint

There are many tasks in which some of the solutions are not acceptable as a result, they are **unfeasible**. Constraints can significantly increase the complexity of a metaheuristic, because every time a new solution is generated (through breeding, mutation and crossover) it has a chance of it being unfeasible.

There are two kinds of constraints: **hard** and **soft**, but their definitions are not necessarily consistent in the literature.

In some works, a **hard** constraint is one that is either satisfied or not, whereas a **soft** constraint is a quality measure that needs to meet a threshold. In other words, a soft constraint is one where some solutions are “closer” to being feasible [61].

In other works, the terminology defines how they are dealt with: hard constraints are those where unfeasible solutions are immediately discarded, whereas soft constraints are those where unfeasible solutions can still be kept if their fitness is promising or if they are closer to being feasible than other solutions (which relies on the aforementioned “closeness” to feasibility). In other cases, a soft constraint is a preference but not a requirement [20]. There are many techniques to represent and enforce constraints in a metaheuristic, with [167] providing a comprehensive overview.

#### 2.3.6.6 Niching / Quality diversity / Crowding / Diversification / Survival / Fitness Sharing

Niching is a less common operator which aims at improving the **diversity** of the population, and is generally associated with multi-objective problems [185]. This simple idea has some subtleties and can be implemented in a variety of ways, which has led to the many names, as seen above.

The differences between the terms for niching are beyond the scope of this thesis. An overview of the meaning of each term is found in [106].

**Diversity** of population is a desirable quality, because it indicates that the algorithm is exploring different parts of the search space (the main purpose of having a population), and niching operators aim to incentivise the population to be more diverse by penalising “redundant” individuals [176].

Niching can be seen as a highly specialised truncation selection method, although it is also used as a tie-breaker in situations where solutions are equally good (more specifically, neither Pareto-dominates the other in multi-objective problems) [72].

There are 2 main ways to categorise niching methods: spatial and temporal.

- **spatial niching methods:** solutions are compared against the rest of the population they are in. These include island models, certain kinds of co-evolution, fitness sharing and even mating restriction rules to prevent “inbreeding” [106].
- **temporal niching methods:** solutions are compared to solutions that were generated in the past (previous iterations or runs). The simplest approach is **sequential niching**, where in a run (or generation) the solutions are stored and any future solutions resembling them receive a fitness penalty or are removed immediately [106].

Moreover, one needs to define what it means for solutions to be similar, which could be done in different ways [106]:

- **in solution space:** based on genotype,
- **in objective space:** based on objective values,
- **in phenotype space:** based on how similar the solutions are in another representation.

### 2.3.7 *Single Objective methods*

This section will discuss and define a range of single-objective metaheuristics which are relevant to this thesis. The most relevant are Genetic Algorithms (GAs) and Simulated Annealings (SAs), with others appearing in [45].

#### 2.3.7.1 *Random Search*

This is the simplest of all approaches, as it consists simply of generating random solutions and keeping the best one. This approach performs surprisingly well considering its simplicity, and it is a good test case as a method which can never converge onto solutions [185].

A similar approach is **random walk**, in which a chain of solutions is generated by starting from a random individual and progressively applying mutations without restriction [185].

#### 2.3.7.2 *Hill climbing*

This approach, in its simplest form, is a heavily exploitative approach which performs best in fitness landscapes where almost all local optima are global optima. The algorithm stores a solution to be the current best, and its initial value is either randomly generated or picked manually. The algorithm then repeats these simple operations [185]:

1. check if termination criteria is met; if so, then terminate.
2. generate a new solutions  $s'$  by applying perturbation on the current best solution.
3. evaluate  $s'$
4. if the fitness of  $s'$  is better than that of the current best, then set the current best to be  $s'$

There are countless variants and improvements to this approach (and one may say that the entirety of metaheuristics is), most motivated by a major fault of Hill Climbing: once a local optimum is found, it is unable to escape its neighbourhood [185].

### 2.3.7.3 Simulated Annealing

Random walk is a method that randomly transitions between solutions, whereas Hill Climbing only transitions to a new solution if it is an improvement from the current solution (in terms of fitness). Simulated Annealing (SA) represents an intermediary between the two: it usually only accepts improvements but occasionally accepts counter-productive mutations [185]. The probability of this event is based on a parameter  $t$ , the *temperature*. For higher values of  $t$ , this probability is higher. Considering a solution  $R$ , and its mutated form  $S$ , the accepted solution is determined by Algorithm 7.

---

#### Algorithm 7 Acceptance Criterion in Simulated Annealing

---

```

1: function ACCEPTANCECRITERION( $R, S, t$ )
2:   if  $F(S) < F(R)$  then
3:     return  $S$ 
4:   else
5:      $p \leftarrow \exp\left(\frac{F(R)-F(S)}{t}\right)$ 
6:     if  $\text{random}() < p$  then
7:       return  $S$ 
8:     else
9:       return  $R$ 
10:    end if
11:  end if
12: end function

```

---

This method is relevant to this work since it is used by BT to solve their SR problem (using iRoster), specifically because it is designed to remain within a small search space region in its final stages rather than producing a less predictable trajectory [81].

### 2.3.7.4 Population Based methods

As mentioned before, many if not most metaheuristics are population-based. There are 4 main categories of population methods, described via the parameters  $\mu$  and  $\lambda$  [185].

- $\mu$  represents the quantity of individuals in the population,
- $\lambda$  indicates the quantity of new individuals generated per iteration.

The first category of methods is  $(\mu, \lambda)$ , where the entire parent population is replaced by a new set of children:  $\lambda$  children are generated, and the  $\mu$  best are kept via selection. The issue with this method is that good solutions can be permanently removed if no good offspring are generated from them [185].

To avoid this,  $(\mu + \lambda)$  methods consider both the previous generation and their offspring when creating the new generation, although only the top  $\mu$  individuals are kept (again via selection). This introduces “competition” between parents and offspring, and allows for a good solution to remain in the population for multiple generations, which is called **elitism**.

The  $(\mu + \lambda)$  GA is the most commonly used metaheuristic within this thesis. The full procedure is shown in Algorithm 8.

These methods have the potential to completely replace the population at every generation, which can be undesirable. As a solution, **steady-state** methods only introduce and remove individuals from the population one at a time [185].

$(1, \lambda)$  indicates a method where a solution will be replaced by one of its offspring, whereas  $(1 + \lambda)$  indicates a method where a parent competes against its children [185].

### 2.3.7.5 Tabu Search

A common challenge in metaheuristics is balancing the preservation of good solutions with the need to explore new regions of the search space. Retaining strong individuals helps maintain quality, but it can also cause the search to stagnate around a local optimum.

---

**Algorithm 8**  $(\mu + \lambda)$  Genetic Algorithm

---

```
1: function MUPLUSLAMBDA GA( $\mu, \lambda$ )
2:    $\text{pop} \leftarrow \text{random\_solutions}(\mu)$ 
3:   function MAKE_CHILD
4:     return  $\text{mutation}(\text{crossover}(\text{select}(), \text{select}()))$ 
5:   end function
6:   while not  $\text{budget\_exceeded}()$  do
7:      $\text{offsprings} \leftarrow [\text{make\_child}() \text{ for } i = 1 \dots \lambda]$ 
8:      $\text{pop} \leftarrow \text{top\_n}(\text{pop} \cup \text{offsprings}, \mu, \text{key} = F)$ 
9:   end while
10:  return  $\text{max}(\text{pop}, \text{key} = F)$ 
11: end function
```

---

In **Tabu Search** this issue is addressed by incorporating a short-term memory structure known as the *tabu list*. Recently visited solutions are recorded, and the algorithm is prohibited from revisiting them for a fixed number of iterations. This restriction encourages exploration and reduces the likelihood of “cycling”.

In practice, many implementations store not entire solutions but rather the moves (e.g., specific mutation or modification steps) that led to them [185].

### 2.3.8 Estimation of Distribution Algorithms

Earlier in this chapter, metaheuristics were described as a “sophisticated form of trial and error” due to how they derive good solutions by applying random changes and applying selection, but the algorithm could have more control over how new solutions are generated. By analysing previously generated solutions, it is possible to build a statistical **model** in such a way that it can be used to generate solutions which are likely to be optimal.

This is the core idea of Estimation of Distribution Algorithms (**EDAs**), although the name **probabilistic model building genetic algorithms** might be better suited. The name Estimation of Distribution Algorithm comes from how the techniques used in these algorithms were originally used to sample data from a given distribution where only the probability density function was available [215].

**EDAs** are not a central topic to this thesis, but they are relevant for their **explainability** potential, as discussed in [14]: by building a model which understands what makes a solution desirable, we obtain information that can be presented to the user for new insights.

The minimal requirements for an **EDA** are [123]:

- an internal model
- a method to build or update the model by using evaluated solutions
- a method to generate solutions from the model

The first significant paper to implement an **EDA** for optimisation was “Removing the Genetics from the Standard Genetic Algorithm” [17], which introduced the relatively simple Population Based Incremental Learning (PBIL) procedure. The current state of the art of **EDAs** lies in Bayesian Sampling (hBOA and variants) [123, 170].

#### 2.3.8.1 Linkage Learning EDAs

A feature that might be of interest for explainability is **linkage**, which is discussed in detail in Chapter 7. In short, it indicates which variables should be considered in conjunction when trying to understand the problem. Linkage information is also useful in a metaheuristic because it allows the refinement of search operators, and thus many **EDAs** are the main source of linkage learning research [295].

**GOMEA** **Gene pool Optimal Mixing Evolutionary Algorithm** (**GOMEA**) and the concept of **Optimal Mixing** are highly relevant to this thesis, since they were the inspiration behind the conception of **PSs**. The idea of Optimal Mixing was first introduced

in [266] where “mixing” is a more general term for crossover, and the algorithms discussed in the paper aim to understand which sets of variables are linked so that they can be inherited” in conjunction during crossover.

GOMEA determines what is the best way to partition the variables of the problem in disjoint sets, which are called **families of subsets** (FOS). For a problem with 5 variables, a Family of Subsets (FOS) might be  $\{\{2, 3\}, \{1, 4, 5\}\}$  [265].

Using FOS information, the Optimal Mixing crossover operator is implemented. These ideas are used later in the thesis, but there are two aspects that should be kept in mind for future reference:

- there is a **metric to evaluate how strongly a set of variables is linked**, but here it is a slow approach that is done once per generation,
- linkage strength is measured by analysing the frequency of each combination of values in the decision variables, and **does not take into consideration fitness values**.

DSMGA / DSMGA-II **Dependency Structure Matrix Genetic Algorithm** (DSMGA) is similar to GOMEA in that it uses linkage information to improve search performance [136]. In short, DSMGA calculates how strongly a set of variables is linked by using *pairwise* mutual information.

The following aspects will be relevant to later discussions:

- the strength of linkage of a subset is based on pairwise linkage, which is much faster than GOMEA’s approach
- Linkage is measured using mutual information, but it still ignores fitness and values like GOMEA

### 2.3.9 Multi-Objective methods

In order to solve a multi-objective problem, a Multi-objective (MO) algorithm must be used.

The main challenge is implementing methods that do not prioritise one objective over another (although that is sometimes the aim), whilst also finding a comprehensive Pareto front [58].

It is important to remember the meaning of **dominance** and **Pareto optimality**, discussed in Section 2.1.4.3.

#### 2.3.9.1 Weighted sum of objectives

Considering the complications brought forward by dominance, there are various approaches to determine if a solution is better than another (**scalarisation methods** [58]), generally by getting a **weighted sum** of the individual objectives.

This method is rather simple because it allows a multi-objective problem to be solved as if it were single-objective, but suffers from significant limitations [187]:

1. **defeats the purpose of multi-objective search**: using a single objective will inevitably introduce biases towards certain objectives.
2. depends on **a priori knowledge of the weights to be applied**.

#### 2.3.9.2 NSGA-II

The Non-Dominated Sorting Genetic Algorithm (NSGA) family of MO metaheuristics is composed of three main members: NSGA, NSGA-II and NSGA-III, with NSGA-II being the most widely used and most relevant to this thesis.

The family of NSGA algorithms started from a  $(\mu, \lambda)$  strategy, used a slow non-dominated sorting algorithm and implemented a more cumbersome niching method based on fitness sharing [299]. This method and the following in the family work by partitioning a collection of solutions (the population) into “Pareto front layers”, which are effectively a **ranking**. The top layer is the true Pareto front, and takes precedence over the rest of the collection; the solutions within the front are considered equally as desirable. Although the solutions in the population do not follow a total order, the

Pareto fronts they form can be ordered from best to worst (i.e. a total order), and thus membership to them can be treated as a fitness value.

When generating a new population, a set of solutions (generally parents + offspring) is first partitioned into Pareto fronts and then these are added from best to worst to this new population until the desired quantity is obtained. The fronts might not fit the required population size exactly, which is where **niching is implemented**. Niching is a technique that serves as a tie-breaker to decide which non-dominated solutions should be kept in the population.

**NSGA-II** introduced the current Fast-non-dominated-sorting algorithm which allows for part partitioning of the population in the Pareto fronts. This second iteration also introduces a  $(\mu + \lambda)$  *inspired* strategy making it **elitist** [299].

Finally, **NSGA-III** was developed to better handle **many-objective** problems (between 4 and 12) by using a more sophisticated crowding measure based on reference points, which is better suited for continuous problems [76]. Due to this specialisation, there is evidence to show that its performance on combinatorial problems is implementation dependent and often worse than **NSGA-II** [111, 141].

There are many other variants of **NSGA** (especially since official source code was rarely distributed for any of them [111]), but the popularity of **NSGA-II** is undisputed due to its relative simplicity and flexibility.

**NICHING** **NSGA** implements a form of **niching**. When there is a set of mutually non-dominating solutions, and not all of them can be kept, there needs to be a way to select which solutions are worth keeping. This is determined by which solutions are more “**crowded**”. Each solution will have a set of solutions it is closest to (in fitness), and the crowding is just the distance between their fitnesses. More information about crowding can be found in Section 2.3.6.6.

Is it felt by the author that there are many cases in which the niching method implemented by **NSGA-II** is not as significant as originally intended. This is because it is a tie-breaker for a set of solutions, and it has been shown to be often as good as a coin toss [299]. Moreover, crowding is sensible for mathematically smooth fitness functions but is ill-suited for rugged fitness landscapes.

### 2.3.9.3 *SPEA2*

Strength Pareto Evolutionary Algorithm 2 (**SPEA2**), introduced in [301], is an enhanced version of the original **SPEA** algorithm, which will not be discussed here. Like **NSGA-II**, it employs a population-based approach and maintains a set of non-dominated solutions across generations. However, **SPEA2** introduces an **external archive** to store elite individuals and uses a **strength value** to evaluate fitness. Each individual’s strength reflects the number of solutions it dominates, and a **density metric** based on k-nearest neighbours is used to implement niching [301].

The algorithm is more complex in implementation than **NSGA-II** and generally runs slower due to the use of kNN and the upkeep of the archive, but finds use in spatial-oriented tasks.

### 2.3.9.4 *MOEA/D*

While it is not suitable to replace a multi-objective task using a weighted fitness function, it is possible to use many differently weighted fitness functions.

The Multi-Objective Evolutionary Algorithm based on Decomposition (**MOEA/D**, introduced in [297]) **decomposes** a multi-objective problem into a set of scalar sub-problems, where each problem is a weighted sum with different weights. Additionally, niching is implemented via a neighbourhood-based selection algorithm.

This approach allows for more control over the “density” of the Pareto front, but does so by using more domain-specific information (knowing what the best and worst fitness values should be).

### 2.3.10 Source of mistrust

Within the field of computer science, metaheuristics occupy the difficult position of not having guaranteed results (unlike traditional algorithms and heuristics) and also converging slowly (unlike Machine Learning approaches).

This chapter aimed to provide evidence for their utility and to give an appreciation of their diversity. On the other hand, it is worth discussing why there is some mistrust in these methods from some audiences.

The following quote from “The Algorithm’s Design manual” [248] provides the point of view of someone who sides more with heuristics:

The idea behind Genetic Algorithms is extremely appealing. However, they just don’t seem to work as well on practical combinatorial optimisation problems as Simulated Annealing does. There are two primary reasons for this. First, it is quite unnatural to model applications in terms of genetic operators like mutation and crossover on bit strings. The **pseudo-biology adds another level of complexity** between you and your problem. Second, **Genetic Algorithms take a very long time** on non-trivial problems. The crossover and mutation operations typically make no use of problem-specific structure, so most transitions lead to inferior solutions, and convergence is slow. Indeed, the analogy with evolution- where significant progress requires millions of years-can be quite appropriate. I will not discuss genetic algorithms further, except to discourage you from considering them for your applications.

This quote presents many valid criticisms, which are presented mainly for Genetic Algorithms but apply to the wider concept of metaheuristics:

- the biological metaphors add a layer of indirection which causes both reduced performance and lack of understanding,
- slow convergence, caused by the lack of specialisation of the methods,

These issues also interfere with the understandability of the resolution process [124]: *The interpretability of Evolutionary Algorithms is hindered by the inherent randomness, dynamics, and population-based mechanisms they employ.*

Because these methods are not exact or easily understandable, the comparison with ML is inevitable. Due to the popularity of ML ( “The development of EC in real-world scenarios is far from satisfactory compared with Deep Learning” [124]), it is sometimes assumed that ML always outperforms metaheuristic methods, and this assumption has the negative consequence of reducing the adoption of metaheuristics in contexts where they were the most suitable choice of method.

One example is a paper from Google regarding automated design of chips [197], where Reinforcement Learning is used for the task despite the task being more suited for metaheuristic methods. A rebuttal ([52]) shows how Simulated Annealing is able to outperform the Reinforcement Learning approach while using fewer resources.

Another example is an article from MIT [88] (which is a discussion of [287]), where the fact that a metaheuristic is able to beat Deep Learning is met with surprise. The following quotes can be found in the article:

- “With all the excitement over neural networks and deep-learning techniques, it’s easy to imagine that the world of computer science consists of little else”
- “But in the last few years, this area of computer science [Evolutionary Computation] has had to play second fiddle to deep-learning machines and their huge success”.

The public’s opinion on metaheuristics is not presented here to be rebuked, but rather to document what causes mistrust and lack of adoption from users. Knowing these reasons, explanations can be developed to make users trust these methods, but also to promote their adoption in the contexts where they are most suitable.

The following provides a summary of the elements of metaheuristics which will be taken into consideration that are relevant to trust, understandability and adoption:

- Although they are simple in concept, using them effectively **requires knowledge on how to choose the right algorithm, operators and parameters**. It is unlikely for a general metaheuristic user to have sufficiently comprehensive knowledge, and therefore the user will be unsure of their setup.

- There is a **trade-off between the simplicity of the metaheuristic and its ability to solve the problem effectively**, due to the No-Free-Lunch theorem. The user is forced to pick a trade-off, but by doing so they disregard more elegant or more efficient methods, thus the user does not trust their own choice.
- Metaheuristics are **perceived as computationally slower than heuristic solvers** and with worse quality of results, but arguably that is an unfair comparison. On a large class of problems, heuristic solvers are inapplicable or do not run within human timescales, whereas metaheuristics are still able to find satisfactory results (even if these are not optimal).
- **Stochastic behaviour**, which causes the quality of the results to be partially determined by aspects which cannot be controlled. If the results change on every run, the user is unlikely to trust that the final result is the global optimum.
- **Excess of information**. A metaheuristic works by generating many solutions, through a mix of logic and chaos. Even by looking at this information, it is both large and mostly unstructured, preventing any human from understanding it.

### 2.3.11 Schema theory

Schema theory was introduced in 1975 by John Holland, as a theoretical foundation to explain the performance of **GAs** [286].

The theory revolves around the concept of schemata, which are also called **building blocks**. These are patterns present in solutions, and they are similar to **PSs** in this sense. The general idea behind the theory is that a **GA** is a procedure which, as it evaluates solutions, also evaluates building blocks and combines them to form new solutions. This framework provides a possible approach to studying the convergence properties of a **GA**, and was influential in some theoretical results in the field (such as uniform crossover being preferred over 2-point crossover) [286].

The topic of schema theory is **controversial**, due to some papers which stated that the entire theory is contradictory and not relevant in explaining **GAs**. Although there has been extensive work in rebutting these accusations, the topic of schema theory is now generally met with distrust. A useful quote by Goldberg from [254] is:

There is no real question that problems contain “building blocks”. The only point of contention is whether **GAs** functions by working with building blocks, or possibly fails to function because it does not work with building blocks.

This small section will give a quick overview of schema theory, due to its resemblance to Partial Solutions (**PSs**). In summary, there are significant differences between **PS** and schemata, and schema theory as a whole is a useful topic.

#### 2.3.11.1 Theorems and Definitions

Considering a binary optimisation problem, where the solutions are of length  $L$ , we have the following terms:

- A **schema** is a string of length  $L$  from the alphabet  $0, 1, *$ , where the  $*$  is the “wildcard”.
- variables in the schema that are not  $*$  are called **fixed**. The quantity of fixed positions is called the **order**.
- A solution **matches** a schema when it contains it.

Schema have also been adapted for other contexts, chiefly in Genetic Programming where a schema represents a subtree of a program.

A **building block** is a high-quality schema, meaning that it is *small* and with *above average fitness*. For a building block, the **average fitness** is the average of the fitness values observed in the solutions which match the building block.

There are then two important hypotheses, which are often misinterpreted [286]:

- **Building block hypothesis:** in a GA, crossover combines building blocks of high average fitness. It does so hierarchically, by first building smaller ones and progressively combining them into bigger patterns.
- **Static Building Block Hypothesis:** A GA is expected to converge to the schema with the best average fitness.

#### 2.3.11.2 *Controversies and resolution*

The paper “The Building Block Fallacy” [268] states that the Building Block Hypothesis is contradictory due to the phenomenon of *epistasis* (feature interactions).

The summary of these criticisms is that there is a contradiction within the Building Block Hypothesis:

1. It is assumed that epistasis has a negligible effect, so low-order building blocks (small beneficial patterns) are found by the GA in its initial stages.
2. It is assumed that there is sufficient epistasis, so that the combination of low-order patterns yields high-fitness high-order patterns, and the GA will then find them.

Additionally, the Static Building Block Hypothesis is far too strong, and it has been shown empirically to be unlikely [286]. Other issues that were raised in [268] were the inapplicability of schema theory, especially since the results consist of probabilities that are only relevant “one generation ahead” rather than for the lifetime of the GA [286].

The two hypotheses were also used to show that crossover is an essential component of GAs, but [199] showed how crossover can have negative effects on search performance.

In particular, they showed how building blocks are found by the algorithm, but they are also lost through random events, or they are associated with underperforming schemata through **hitchhiking** [199].

This is the phenomenon where a genetic trait spreads throughout a population due to chance rather than because it contributes positively to fitness. This is caused especially by its coincidental co-occurrence with a positive trait [199].

After the fall from grace of schemata, few researchers continued to work on them, some with the aim of redeeming the value of schemata. The work of these people showed that the criticisms were ill-posed or overly specific. Riccardo Poli is to thank for many developments on schemata in this time period, especially in the context of Genetic Programming. The rebuttal papers are long and dense, whereas the initial criticisms were small and easy to spread, and therefore the general reputation of schemata has never recovered [286].

# 3

## BACKGROUND ON EXPLAINABILITY

---

In computer science, **explainability** refers to a set of features and ideals in response to issues related to **black box** methods.

The Merriam-Webster dictionary gives four definitions for “explaining” [191]:

- to make known
- to make plain or understandable
- to give a reason for or cause of
- to show the logical development or relationship of

These definitions have an intuitive meaning for humans, but when trying to apply them in the context of computing, they can be hard to define precisely or implement.

**THE BLACK BOX PROBLEM** When a human solves a task, their approach can be understood by asking the human to provide an explanation behind their thought process. When these tasks are solved by a computer using an automated process, it is sometimes impossible to obtain an explanation. The term **black box** refers to the inability to understand a process, due to it being hidden or practically impossible to understand [121].

**RAISON D’ÊTRE OF EXPLANATIONS** Although using a black box algorithm can solve the original task at hand, it also introduces many issues. These include ethical concerns, legal issues, and matters of practicality. The field of **explainability** was born as an answer to these issues and it is inherently multi-disciplinary.

The study of explanations - **XAI** - is the composition of many different ideas (as shown in Figure 3.1), with this quote summarising it well [40]:

XAI is an umbrella term for any research trying to solve the black-box problem for AI. Since there are a lot of different approaches for solving this problem, each with its own individual needs and goals, there is no single common definition of the term XAI.

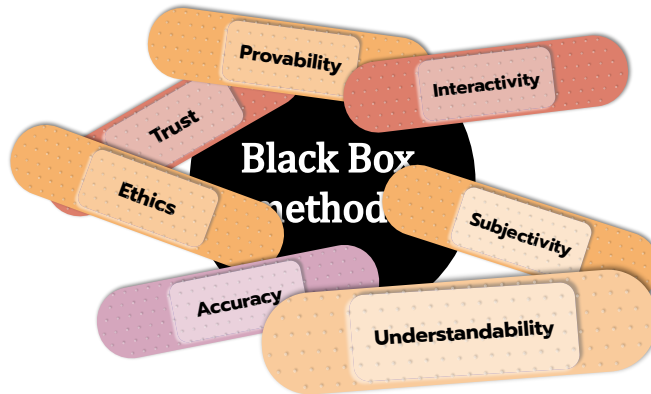


Figure 3.1: This figure represents explainability as many plasters attempting to resolve the issues caused by black box methods. These many aims often overlap.

**STRUCTURE OF THIS CHAPTER** This chapter will first give a **history** of explainability in Section 3.1, especially since many of the motivations and terminology are retrospective. We then present essential definitions, terminology and nuance in Section 3.2. The **motivations** for **XAI** are then discussed and compared in Section 3.3. This is followed by various definitions and characterisations of explainability in Section 3.4. A state-of-the-art review of **XAI** is given in Section 3.5, supplemented by a discussion of the intersection between **XAI** and **metaheuristics** in Section 3.6. Finally, the chapter ends with a discussion of the issues and challenges within the field in Section 3.7.

### 3.1 HISTORY OF EXPLANATIONS

**BEFORE XAI WAS A TERM** Explainability as a feature can trace its roots to **early AI systems**, since their simpler structure (based on symbolic logic) meant that their internals were inherently human-understandable, with relevant papers appearing as early as the 1970s and 1980s [114]. A term that was coined at the time was **expert system**: an automated computer system which can make decisions based on known rules [260, 294].

These kinds of models have stood the test of time: decision trees, linear regression and Learning Classifier Systems are all products of this time period.

**TRADING UNDERSTANDABILITY FOR ACCURACY** Transparent models have over time been overshadowed by **black box models** in **ML**, namely Deep Neural Networks and their derivatives such as Transformer Architectures. These methods are generally thought to have increased predictive performance at the cost of understandability, which would suggest a **trade-off** between quality of results and understandability [130]. This trade-off was assumed to be factual in early explainability research, but more recent research has shown it not to be universally true [232]. Research into how **ML** methods can become understandable again became known as Interpretable Machine Learning (**IML**), which was a short-lived name due to the rise of the term **XAI** [284].

**XAI, AND DARPA** The topic of **IML** was conceived as a subset of **ML**, but the concept of explainability quickly outgrew **ML** and became what is now known as **XAI** around 2015 [284]. Originally, **XAI** covered a larger topic than **IML**, including metaheuristics, interpretable agents, AI planning and Human-In-The-Loop methods [3, 42].

The **Defence Advanced Research Projects Agency (DARPA)** crystallised many of the concepts of explainability with its **XAI** program "DARPA-BAA-16-53", which ran from 2017 to 2021. This program delineated three research interests [84, 114]:

1. development of new **XAI** for **ML**
2. application of **psychology** to produce theories of explanation
3. evaluation of **XAI** techniques in data analytics and autonomy

The research directions produced as part of the DARPA project were strongly skewed towards **ML**, which led to the term **XAI** having a strong association with **ML** despite its intended generality [284]. This issue is likely exacerbated by the ever-shifting meaning of Artificial Intelligence (**AI**) [143].

### 3.2 ESSENTIAL TERMINOLOGY

There will be many terms used throughout this chapter, but there are some that are particularly common and often misused. Some terms to be careful about are **explanation**, **explainability**, **XAI**, **interpretable**, **understandable** and **transparency**. The misuse of these terms is well known within the field of **XAI** [182, 34]. Other definitions and discussions of the terms can be found in [12, 182, 221, 276].

**EXPLANATION** Firstly, it is necessary to define the term "explanation".

For an algorithm or computational process, an **explanation** is the result of offering information or content to a person. This interaction promotes **human-centred** properties (such as understanding, increased autonomy, or trust) rather than contributing directly to computational efficiency or quality of results.

The above statement is crafted carefully to provide the lowest common denominator across explainability literature. Surveys on the topic often focus on specific aims such as trust and subjectivity, with their common trait being the **human element** [231]. Defining an explanation as “the result of giving information to a user” is adapted from [284], where it is argued that other definitions exclude aspects such as subjectivity and non-ML methods.

## EXPLAINABILITY AND XAI

The terms **explainability** and **XAI** will be used to refer to the **research topic** of producing explanations for computer systems, including metaheuristics.

The term “explainability” has been used to describe models that can be made clear through explanations, but this definition has been used inconsistently and has been criticised for not aligning with scientific rigour [212]. To avoid confusion and mutually recursive definitions, this meaning of the term is avoided here; the term is only used to refer to the *research field* of explanations. As mentioned in Section 3.1, the term **XAI** is also used inconsistently and is also used to describe the same topic, which for this thesis will include explanations for metaheuristics.

## INTERPRETABLE, UNDERSTANDABLE AND TRANSPARENT

A piece of information is **understandable** if its meaning is understood by the relevant audience with little effort. For an explanation to be successful, the understandability of information is generally required.

The literature uses the term **interpretable** to mean that an explanation can be understood by a human [82]. However, this term has also been defined to mean a system which can be inspected [94, 276]. To avoid confusion, the term **understandable** is used instead, since it is less ambiguous [212]. The use of this term is paired with the notion that different humans understand things differently.

The term “transparent” is often confused with the previous two terms. In [19] it is stated that “a model is considered to be transparent if by itself it is understandable”, but in [276] it is used to mean “the capacity of a method to explain how the system works even when it behaves unexpectedly”. These are two sides of the same coin, and the following definition is used.

A model or piece of information is **transparent** if it requires **no additional information** in order to be understood and to result in explanations.

## 3.3 MOTIVATIONS

Intuitively, explanations are desirable because they are “value added” to an automated procedure. This value comes directly from how humans intend to use the system, and can take many forms.

There are many desiderata for explanations, and this section will attempt to give a representative sample of what is presented in the literature. Each of the motivations shown has been used as a definition of explanation in the literature.

Many different uses for black boxes lead to different issues, different needs and different methods of satisfying them, resulting in a collection that does not lend itself well to classification [34, 40].

**CAUSABILITY, ANSWERING “WHY?”** Generally an explanation needs to convey useful and accurate information; an explanation should give reasoning or, more simply, answer the “**why**” questions and present the “reason-for or cause-of”, which has been termed **causability** [276, 85].

What the cause-effect relationships look like as explanations can vary greatly: in old logic-based systems (such as Prolog) there is a sequence of logical statements that is

essentially a proof, whereas in a Neural Network for image classification it might be a heatmap of which pixels most contribute to the prediction [294]. For some, the latter does not count as causability since they do not provide a full answer to how certain information was used [94].

The question of **correlation vs causation** is also relevant here, as information for explanation might propose correlations in order to suggest *plausible* causations rather than provable reasons [232].

**LEARNABILITY** For an automated method which solves a task better than humans, there might be interest in having the machine teach humans the process that it learned. This aspect has been called **learnability** in the literature [276]. In this sense, explanations can be thought of as a method for information extraction and compression: a compact ruleset for a complex system is automatically found by the computer [14]. Often this ruleset is a “lossy” compression, where some information is lost for the benefit of understandability.

**UNDERSTANDABILITY** The internal structure of a Neural Network might be open for study, but it is often not humanly comprehensible, whereas a small set of simple rules might make sense to a human. The property of understandability is of interest because it is lacking in **black box** systems.

Understandability is the subject of endless redefinitions, due to its context-dependent nature; the ability of a user to understand an explanation is influenced by the **background of the human, time available, complexity of the subject, quantity of data**, etc [12]. The question of what makes information understandable is inevitably rooted in cognitive psychology: the **cognitive load** required to process a piece of information affects the user’s ability to process it, their user experience and their overall performance at a relevant task [129]. An explanation may be seen as a tool to lower the cognitive load involved in understanding information [19].

**ETHICS** The aims described above are themselves caused by more abstract aims, many of which stem from **ethics**. The nature of ethics is broad, but [206] gives 3 relevant frameworks:

- **Duty**: explanations can help respect moral duties, such as **truthfulness** and assisting **autonomy**
- **Societal impact**: the impact which a black box system has on society might be unclear, such as when there are biases in decisions that affect people [232]. Explanations can help in avoiding negative societal consequences.
- **Virtue ethics**: explanations may assist in the development of good characteristics in humans, such as honesty, benevolence and wisdom.

**LEGALITY** Legality is the manifestation of the ethical considerations brought by the use of algorithms and especially black box models.

The European Union was the first to underline the importance of certain rights as part of the “automated decision-making” section of the General Data Protection Regulation. Many smaller rights are clustered under the name “right to explanation” [21]:

- the right to obtain **meaningful** information about a decision,
- the right to **contest** a decision,
- the right to human **intervention**.

These concepts are mainly derived from the ethical aims of honesty and autonomy. These were then formalised further in the **Artificial Intelligence Act**, which relates to high-risk systems such as banking and healthcare. The Act describes how the materials provided by explanations must be sufficient to allow **contestability**: the **challenging** of automated decisions [205].

Similar rights are granted by China’s “Personal Information Protection Law” [193]. In Brazil the “Lei Geral de Proteção de Dados” is similar but leans more towards **fairness**: an automated system needs to be transparent enough that any discrimination or bias is known and avoided [193].

Similar purposes are pursued at a smaller scale by private companies, where explanations are used to ensure legal compliance, traceability and auditing [14].

**TRUST** DARPA’s **XAI** program states three main aims of explanations: to understand models, to **trust them** and to better **manage them** [84]. In **sociological** terms, a human trusts an automated method when: (1) the user is *vulnerable* to the actions of the computation, and (2) the human is able to *anticipate* the outcome of the computation [145]. Trust is generally conceptualised in four ways [221]:

- **quantifying confidence and knowledge limits**: the results of the systems are accompanied by confidence intervals, proofs of correctness, and other materials that quantify how much the automated method should be trusted,
- **showing that the process follows a sensible “thought process” that aligns with human intuition**: humans see that the computer follows a sensible procedure which they can understand and therefore trust as they would trust themselves,
- **the system has no unknown biases or loopholes**: the system can be deployed without worry for unintended biases or possibility for **adversarial** attacks,
- **showing that the process has satisfactory results**: accuracy, reliability, security, privacy etc.

Another aim is to promote **wider adoption** of automated methods, where this is hindered by a lack of trust [290, 14].

This interest in trust is born out of the aforementioned sociological requirements: black box models are used for critical applications (users are *vulnerable*), and their internal workings are hard to mentally simulate (so that they cannot be anticipated) [145].

Considering trust as the sole objective leads to various issues; a user can be tricked into trusting a biased system [94]. Moreover, user trust is not necessarily a result of user understanding - the user might *think* that they understand a topic, but in reality they do not. This effect, termed the **illusion of explanatory depth**, is a naturally occurring phenomenon in humans [55]. For some, explanations are a tool to investigate the possibility of adversarial attacks, both to avoid them and to produce them [18].

**SUBJECTIVITY** How information is processed depends on the user and their context. Different users have different needs, and different **mental models** [192]. In the same vein, it cannot be assumed that an automated process can be understood universally, and many researchers view explanations as a tool to tailor information to match the needs of a user. This tuning could take many factors into consideration (including feedback and new information), with the final aim of improving the system’s effectiveness and keeping it relevant [290].

**PRACTICALITY AND DECISION SUPPORT** Beyond the many aims discussed above, explanations may also be sought due to **practical advantages**. The presence of explanations might assist a decision-making process dependent on automated methods. Practical advantages include [14]:

1. **debugging**: the system might be achieving high performance by exploiting a loophole. Explanations can help in isolating these scenarios by making certain aspects of the system more transparent
2. **malleability**: there are many scenarios where last-minute *manual* changes need to be applied to automatically derive results. Such modifications can be informed by explanations.

The latter point is relevant to the right to explanation in law: judicial decisions based on automated decision making might need to be adjusted manually when disputed [232].

There are various scenarios where the automated system is part of a **trial-and-error** process, such as a metaheuristic, hyper-parameter tuning, or even CRISP-DM (Cross-industry standard process for data mining).

An interesting perspective is given in [39] where **XAI** is discussed for the **arts**: an artistic process is similar to developing and using an automated system in how it is rarely done correctly on the first try and it is inherently a trial-and-error process. Explanations can help users in their iterative process by:

- identifying and formalising the **objective** (which might be subjective)

- making the process of finding a solution more **interactive**
- **identifying issues** in current results

The second point is particularly interesting because it is unexpectedly relevant to this thesis: consider how trial-and-error is involved in optimisation (1) when solving the optimisation problem and then (2) when modifying solutions to implement them in the real world. The points raised by [39] are highly relevant:

From an XAI perspective, the question here is not how to better explain the AI content creation process, but rather how to use XAI to **reduce barriers to trial and error** given that it is inherent in many artistic practices. Supporting trial and error also asks how to use XAI to help identify differences between an artist’s expectation and an AI’s output to help iteratively bring the two closer together.

Compare this against a paper discussing explanations for metaheuristics [14]:

When we define a problem, it is often hard to fully codify all the real-world goals of the system. [...] This also relates to one of the motivating factors behind interactive evolutionary computing - we want something that is mathematically optimised, but also something that corresponds to the problem owner’s **hard to codify intuition**. By incorporating XAI into interactive evolutionary computing we could make it easier for the problem owner to interact with the optimiser.

Having explanations which help produce better results might appear to contrast with the definition of “explanation” given in Section 3.2 anything which improves **human-dependent** aspects. The practical aspects described here are still dependent on human users, and explanations are guidance tools to help humans work on these practicalities.

### 3.3.1 *Motivations specific to BT*

It is useful to see how the aforementioned motivations can apply to a real scenario, such as the Staff Rostering problem encountered by BT. Firstly, the motivations relate to the following stakeholders:

- **workers**: Need to detect and prevent bias in allocation
- **rota manager**: Need to detect and reduce issues which negatively affect efficiency
- **HR manager**: Prove that the metaheuristic used is trustworthy enough to be implemented
- **optimisation researcher**: Debug and verify if the metaheuristic is working as intended, and if the objective is formulated correctly

Then the following motivations are relevant:

- **Causability, Ethics and Law**: workers have the “right to explanation” on why they received certain allocations, especially in comparison to similar workers. The need for explanations is exacerbated by the presence of a **societal impact** on the workers, and the possibility of the algorithm being **unfair** towards certain workers.
- **Learnability**: the task of deciding rota patterns is hard to teach to new managers. Our industrial collaborator has suggested that explanations might be useful as a teaching aid for the metaheuristic-based tool.
- **Understandability**: The solutions to the optimisation problem are complex due to the quantity of variables and the complexity of the fitness function. The main complexity lies in understanding what makes a solution near-optimal. Understandability is a requirement for many of the other motivations, such as trust.
- **Trust**: the algorithm needs to be approved by all parties, which requires trust. According to our industrial collaborator, many metaheuristic-based methods are not permitted by HR due to lack of trust, and there are instances of rota managers ignoring the results produced by the algorithm. The workers themselves need to trust the system which allocates them, for ethical reasons.

- **Subjectivity:** Although there is a fixed fitness function, it is unclear if this encapsulates all of the properties required by the rota manager. There might be implicit objectives coming from the rota manager and the workers themselves. The fitness function contains various magic numbers, which might be an approximation of subjective preferences.
- **Practicality:** For the optimisation researcher, **debugging is relevant** - it is necessary to determine if the fitness function is implemented correctly, especially since there are edge cases that are not well defined. For the rota manager, **malleability** is more important since they often have to modify solutions due to last-minute changes (such as workers being sick).

Both parties participate in an **iterative process** towards making the automated program fit in with many hard and soft constraints deriving from human availability, workers' rights, financial interest, convenience and reliability.

### 3.4 DEFINITIONS AND CHARACTERISATION OF EXPLANATIONS

Explanations are often described with respect to a “model”. A model is a **simplified representation** of a relationship. This definition is discussed in [192], where the model is not always a **ML** model. It could represent why a solution has good fitness, for example.

An explanation is usually a simplification of a model, which makes the explanation itself a model. By describing this latter model, we can categorise explanations.

#### 3.4.1 *A priori and A posteriori*

A useful category to start from is the research directions which the DARPA program described for explainability [114]:

- **A priori = transparent methods:** the method which automates a task is transparent: the prediction model and the explanation are one and the same. In this sense, the explanation is the *cause* of the observed behaviour. This includes linear and regression models, decision trees and other rule-based systems. More generally, it also includes simple heuristics.
- **A posteriori = post-hoc = probing of black boxes:** explanations are generally *descriptions* of observed behaviours, or they are **plausible causations**. Variable importance methods such as SHAP are generally in this category.
- **model induction techniques:** make simpler methods that mimic the behaviour of a black box model. This can apply to many models, and could include LIME.

The last of these three could be considered both an *a priori* and a *a posteriori* method. Therefore, the two labels are not mutually exclusive.

From these short definitions, it would appear that *a priori* explanations are always preferable, and some argue that this is the only kind of explanation [232]. This point of view aligns with the motivation of **causability**, but explanations have many more aims. From the point of view of **practicality** and **decision support**, post-hoc explanations are sufficient in assisting users for various purposes without the need to be perfectly accurate [34].

#### 3.4.2 *Explanandum and Explanans*

An explanation is composed of an **explanandum** and an **explanans** [221]:

- **Explanandum / Object of explanation:** model / process to be explained
- **Explanans:** what aspect of the explanandum is being explained?

These are clear-cut terms for **ML**, but in other fields they are not as clear. In the case of explanations for metaheuristics, the explanandum can be:

- individual solutions (our work)
- a population at a specific generation (work of Fyvie et al, [96])
- the entire population over the entire run (fitness landscapes, [267])
- the search space (local optima networks, [209])
- a surrogate model that predicts fitness ([36, 247, 278])
- search operators (mutation, crossover, selection etc) (IOHXplainer [273])

### 3.4.3 *Format of explanations*

Another aspect of explanations that cannot be ignored is **format**, which describes how the information is conveyed to the human.

Traditional **XAI** relates to **static** explanations, which are generated once and are the same for every user. The information presented can be numeric, visual, or even textual [221].

In contrast, other **XAI** methods have a higher **degree of interactivity** and are thus termed **dynamic**: the user has some control over the explanations being produced.

These can be further split into [221]:

- **declarative**: There is only one question being answered, which cannot be changed
- **one-way**: The user can decide what question is being answered by the explanation
- **two-way**: The user can decide what question is being answered, and the computer can request information from the user

### 3.4.4 *Local and Global explanations*

Another important categorisation is the **scope** of the explanations [11], which is based on the *explanandum*:

- **global** explains the behaviour of the automated process overall
- **local** explains the behaviour of the process with a specific input

These labels align well with the datapoint-dataset dichotomy in **ML**, but they are not universally applicable terms.

This thesis will mainly relate to the Staff Rostering problem, where the difference between local and global explanations is significant:

- A local explanation will generally refer to explanations for the best solution found by a metaheuristic, such as “the most important part of the best solution is...”;
- A global explanation will generally refer to explanations that apply to all solutions, such as “this group of workers is strongly linked”.

### 3.4.5 *Model-agnostic and model-specific*

Regarding their implementations, some explanations are specific to certain models, while others are developed to be more generally applicable. In the case of black box **ML** models, the following terms are often used [12]:

- **model-agnostic**: applicable to any method, and only assumes that predictions can be obtained for a variety of inputs.
- **model-specific**: the explanation information is obtained by aspects which are specific to the structure of the model.

This classification is used in most **XAI** taxonomies, such as the one shown in Figure 3.2.

This thesis does not focus on **ML** models, but readapts this terminology for metaheuristics and therefore uses the terms **metaheuristic-specific** and **metaheuristic-agnostic**.

## 3.5 STATE OF THE ART OF EXPLANATION METHODS

Within the many approaches existing within **XAI**, there are some that stand out in popularity and in results, which will be shown here.

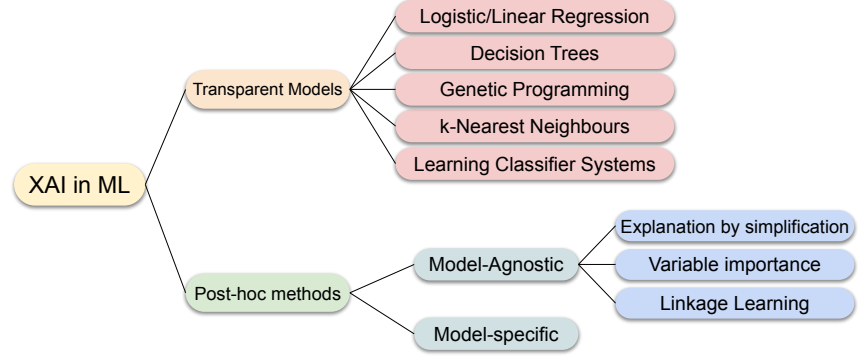


Figure 3.2: Summary of the taxonomy provided in [12], only including the sections which are relevant to this thesis.

### 3.5.1 *Transparent models*

As defined in Section 3.2, a model is **transparent** if it can be used to produce explanations without the need for additional material. This abstract definition is often associated with the idea that certain models are “always transparent”, which is a fallacy [94]. Although the *format* of a model’s internal mechanism is “easy” to understand for a human, the quantity of information present might prevent true understanding of the entire model [45]. Furthermore, the idea of universal understandability clashes with the aim of subjectivity in explanations (presented in Section 3.3).

Therefore, transparency in itself is a nuanced topic. Still, it is still clear that some models are generally better than others in being understandable by humans. It is easier to think of **understandability as a constraint**: certain models are guaranteed to break it (black box models) whereas others can satisfy it if the right conditions are met.

**LINEAR REGRESSION AND LOGISTIC REGRESSION** Linear regression has the advantage of plainly indicating the effect that each input decision variable has on the output [234].

With an input  $x$  which is a vector of  $n$  parameters, a linear model will consist of parameters  $\beta_0, \beta_1, \dots, \beta_n$  (also called coefficients). The parameters and the input vector are combined as follows [148]:

$$f(x) = \beta_0 + \sum_{i=1}^n x_i \beta_i \quad (3.1)$$

The sign and magnitude of the coefficients can be used to determine what effect each individual input has on the output [234]. Obtaining these parameters is the complex part, but it is achieved through **least squares regression** [148]. This process is successful if the data satisfies certain assumptions, and may be tuned to represent the human preference for smaller coefficients (regularisation) [148].

**GENERALISED ADDITIVE MODELS** Linear models can be improved by introducing additional steps between the true input and what is passed to the linear models. The original input goes through an additional “non-linear transformation” step before being passed as the input of a linear model. This allows the overall system to infer non-linear relationships while still maintaining additive decomposability [122].

**DECISION TREES** Decision trees provide hierarchical rule-based explanations which take the shape of an inverted tree, as shown in Figure 3.3.

In order to make a prediction about an input, the tree structure is navigated based on whether the input satisfies the condition at each decision node. Obtaining the optimal decision tree from training data is a complex problem, although algorithms such as C4.5, ID3 and CART are generally thought to be sufficient for most applications [45].

The main appeal of decision trees lies in their understandability: the if-then rules are simple. Additionally, the complexity of these trees is **tunable**: by removing any branches

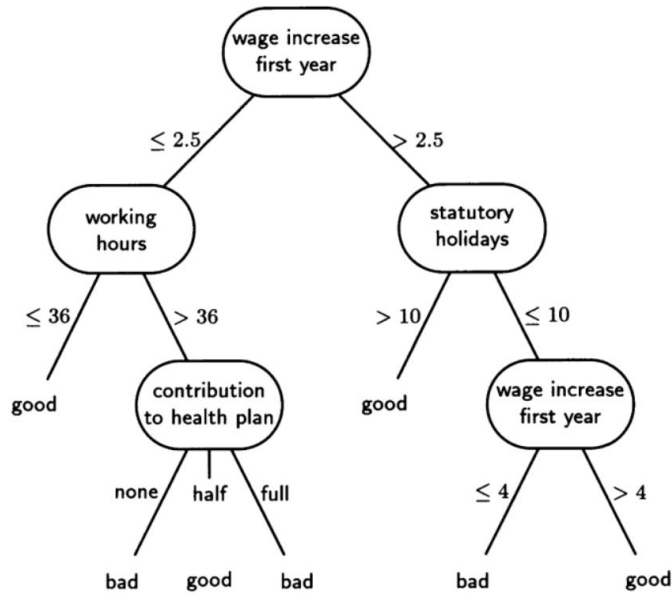


Figure 3.3: Example of a decision tree, as shown in [225]

below a certain depth we obtain a model that is simplified while still retaining most of its predictive accuracy [23].

**GENETIC PROGRAMMING** The tree-like structure used by decision trees is shared by the format of the results produced by Genetic Programming. These are evolutionary algorithms which evolve mathematical formulas or more generally programs [190]. These results are more general than decision trees, since they can express richer operations than if-else rules, and they retain their relatively understandable structure [277].

At the same time, the increase in degrees of freedom also leads to a much more complex (often multi-objective) optimisation problem where predictive power and model size are at odds [277]. The optimisation task is complicated by the tendency for *bloat* to arise: it appears to be evolutionarily advantageous for trees to include large unnecessary subtrees. The emergence of bloat is a considerable obstacle to understandability, as their prevent trees from being human-understandable [190]. Defining how the size of the trees impacts their understandability is possible via empirical measurements, and such formulas have been used in many works [277].

**K-NEAREST-NEIGHBOURS** The models discussed above are inferred from some training data, but that data is discarded once model training is done. A k-Nearest-Neighbour model takes a different approach, where the data *is* the model.

Given an input data-point, the model will determine the k-nearest data-points to it (based on a distance function) and aggregate them to produce an output. At the same time, these data-points serve as the explanation for the prediction [64].

#### 3.5.1.1 Learning Classifier Systems

Learning Classifier Systems are a model consisting of a set of rules, and the process which derives these rules is the progenitor to Genetic Algorithms [272].

The model is composed of a set of rules, each in the form of a rule associated with a prediction. Given an input, the model will first check which rules it matches, and will then aggregate their predictions.

The rules themselves are selected to be simple, and they are similar in format to schemata and PSs. These rules are learned through a stochastic iterative process based on training data [272].

### 3.5.2 Variable importance methods

The coefficients used in a linear model are praised for their direct understandability, so much so that many post-hoc methods were designed to produce similar results for scenarios where a linear model is not used. These form as an introduction to post-hoc methods, and are the most commonly used tool in XAI [294].

**LIME** Locally Interpretable Model-Agnostic Explanations (LIME) pioneered the idea of local explanations [227]. Considering an input, small changes to its parameters lead to small changes in a model’s prediction. LIME finds a linear model that represents this relationship, and presents it to the user.

Additionally, LIME was also one of the first approaches to be explicitly **model-agnostic**, meaning that it is designed to be applicable as a post-hoc method for many kinds of black box models. Although LIME was one of the pillars of XAI at its start, it is now being heavily scrutinised due to its instability [235, 298] and potential for adversarial attacks [249, 168].

**SHAPLEY VALUES** Consider a scenario where there is a set of items, and each subset of them is associated with a certain reward. The inclusion of a certain element might lead to an increase in *expected* reward, and this determines the overall contribution of that element to the overall collection.

Shapley values (which are different from SHAP) (first introduced in [242]) are a mathematical approach to quantify such contributions, and find their application in explaining models because each input variable can be treated as an element that contributes to model performance [289].

In the case of ML, a set of input variables can be used to train models of differing accuracy. The Shapley value of a feature is the (weighted) average improvement in predictive performance when the feature is included vs when it is not included, for any subset of features [289].

Although Shapley values possess many desirable mathematical properties, obtaining them is inconvenient or computationally unfeasible for most ML models. Firstly, it requires a way to evaluate the model with only a subset of the features, which can be done by rebuilding a model every time and applying it. If this route is taken, then every possible set of features needs to be evaluated, meaning that there are  $2^{\text{\#variables}}$  models to be built and evaluated.

Therefore, Shapley values in their pure form are generally not applicable without making assumptions on the model, which is how SHAP is defined.

**SHAP** SHAP is a special form of Shapley values, obtained by assuming that the model in question is **additively separable**, which avoids the need for model re-training [186].

SHAP fits within a framework where a black box model is being mimicked by an additively separable model. The model in use varies significantly between different implementations of SHAP, with KernelSHAP and TreeSHAP being based on linear regression and decision trees respectively [196].

The method makes significant assumptions on the true underlying function, such as the lack of interaction effects and that each variable has a linear effect [186]. Although these assumptions make it theoretically inapplicable to most tasks, these have been ignored and have led to SHAP being the most well-known XAI method. Similarly to LIME, there is a growing body of literature noting various faults with the method [249].

At the same time, it serves as a good example of what people want from explanations: something widely applicable and easy to use. It satisfies the needs of many users (trust, debugging) but it might not be yet sufficient in the aims of causability [94, 34].

SHAP and LIME are considered to be almost equivalent methods (by design of the paper that proposed SHAP [186]), but it has been found empirically that the two are not equivalent and often times in disagreement [164].

**PERMUTATION FEATURE IMPORTANCE** If an input variable is important, then changing its value should have a strong influence on the output. If one wants to quantify how strongly a model relies on a variable for its predictive accuracy, an easy option is to permute the values of that feature over a dataset and measure the decrease in performance.

This method, called **permutation feature importance**, has the advantages of being model agnostic and not requiring any model retraining [10].

#### 3.5.2.1 Sensitivity analysis / Perturbation

Another class of methods which produces variable importance scores is **sensitivity analysis**. This measures how sensitive the output is to small changes to each individual feature (one at a time). In the case of the output being a single scalar value, then it can be defined as the absolute value of the change in fitness when one variable is changed at a time [275].

To note is that this technique does not assume that a machine learning model is present, and it can be applied directly to any function. In these cases, the name **perturbation** is more common (especially if it is related to adversarial examples [16]).

This method is also highly model-agnostic and avoids retraining, which makes it applicable to metaheuristics [173].

#### 3.5.2.2 For Neural Networks

Neural Networks have been a focus of **XAI**, and thus they benefit from many variable importance methods. A summary list is as follows:

- **Saliency maps**: Computes the gradient of the output with respect to the input, and that is used as the importance [246].
- **SmoothGrad**: A technique that reduces noise in gradient-based attributions by adding visual noise to the input image multiple times, computing the gradient for each, and averaging the results [251].
- **Integrated gradients**: calculates the integral of gradients along a straight path from a baseline (e.g., an all-zero image) to the input [259].
- **DeepLift** finds the contribution of each input relative to a reference point [174]
- **Layer-Wise Relevance Propagation**: Back-propagates neuron weights from output to input [200].

#### 3.5.3 Linkage Learning methods

When attempting to understand a function or a model with multiple inputs, human intuition tends to assume that each input has an independent effect on the output [71].

Unfortunately, real-world problems tend to present **interaction effects** between variables: the effect of one variable on the output is dependent on other variables, and thus its overall behaviour is hard to describe [71]. **Linkage learning** aims to find these relationships, and to present them to the user so that they will be aware of a potential underlying structure in the function or model that is being examined [14].

This topic is arguably understudied in the literature, but it is highly relevant to this thesis and it is explored in much more detail in Chapter 7.

### 3.6 EXPLANATIONS AND METAHEURISTICS

Due to the themes of this thesis, it is useful to describe the interplay between metaheuristics and **XAI**. As described in [14], the relationship goes in two ways:

1. **XAI “within” metaheuristics**: the black box being explained is the metaheuristic,
2. **Metaheuristics “for” XAI**: how search methods can be used to generate explanations

This thesis belongs to both categories: a metaheuristic is to be explained, and the explanations are generated using a metaheuristic.

### 3.6.1 *XAI within metaheuristics*

Explainability methods for metaheuristics are diverse, due to the many elements that are involved in a metaheuristic task.

The methods shown here are based on the review proposed in [14] and the more recent review from [1].

**LANDSCAPE ANALYSIS** The main method seen for metaheuristics is **landscape analysis**, which studies the landscape of the fitness function [267]. Within this, there are two research streams: **search trajectory networks** and **local optima networks**. Search trajectory networks summarise how the metaheuristic “moves” between regions of the search space, in terms of transition probabilities [210]. Local optima networks are similar, but they focus on local optima and how the search transitions within them [209]. A similar idea is taken by barrier trees in [116], although the resulting graphs focus more on the fitness values rather than connections between objective space regions. A further discussion of these methods is provided in Chapter 4.

**HUMAN IN THE LOOP METHODS** If an algorithm is a black box because the human is not part of the process, **Human-In-The-Loop** methods put the humans in the box so that its internals are not as obscure. In more practical terms, part of the algorithm is implemented by having the user provide information interactively.

These are often a forgotten branch of **XAI**, since it does not fit within the labelling of a-priori or a-posteriori. A review of these methods can be found in [166].

Human-In-The-Loop methods can tackle a problem in optimisation that is often ignored: the true fitness function is hard to encode (due to aesthetic preferences, multi-objective trade-offs, subjectivity etc), and thus it’s much more transparent to use a human directly as the fitness function or as a search operator (such as selection). This topic usually takes the name of **Inverse Reinforcement Learning** [4].

**METHODS RELATING TO METAHEURISTIC OPERATORS AND HYPER-PARAMETERS** Although a researcher of metaheuristics has a good intuition for which operators and which hyper-parameters to use, their effect can be quantified by the use of explanations.

One such method is **IOH-Xplainer** [273], where explainability methods (SHAP mainly) are used together to give an indication of metaheuristic performance, as well as SHAP importance scores for various operators and hyper-parameters such as population size and mutation rate.

Another method, **EvoMapX** [1], produces various plots which quantify the effect of each operator, but at different stages of execution [1]. **Flacco** [156] is another notable mention, as it allows the user to quickly obtain many metrics relating to the fitness landscape. A survey of similar methods is provided in [1].

**TRAJECTORY ANALYSIS** Trajectory mining focuses on the *sequential* nature of metaheuristics. Metaheuristic performance is often discussed in terms of exploration and exploitation, which ideally should be at the start and end of the search process respectively. In order to quantify whether this is taking place, it is possible to analyse the temporal information of population-based methods. This research is described in [96], where for example the “convergence” of the population is plotted over time. An in-depth discussion is provided in Chapter 4, since much of it is applied to the same Staff Rostering problem as this thesis.

Similar methods focus more on the visualisation aspect:

- **Evo-Panel** [54]: shows the behaviour of the search method by plotting the fitness function as a heat-map and the solutions as dots.
- **Performance2Vec** [86]: representing algorithm performance as embedded vectors, which can be processed for example through clustering.
- **PopDP** [279]: plots population trajectories in multi-objective problems

### 3.6.2 *Metaheuristics for XAI*

The task of finding explanations often resembles an optimisation problem; as discussed in Section 3.3, users and researchers often have an *aim* of what the explanation should

do, but do not exactly know how to achieve it. This challenge can be solved by stating the aim as a function, and solving it through a metaheuristic. Multi-objective search problems are particularly relevant, as there may be multiple conflicting objectives [14].

One example is the evolution of **white box** models: a complex model might be replaced by a simpler Genetic Programming (GP)-tree. In these cases, the objectives are both predictive accuracy but also the complexity of the tree [190], but in other cases even fairness can be used as an objective [14].

Another example is the generation of **counterfactual explanations**: for a given input and its output according to a model, one can find the smallest modification to the input which leads to a different output [243]. A review of the methods can be found in [274].

### 3.7 CHALLENGES AND ISSUES WITHIN THE FIELD

Explainability as a field has a set of complex challenges that are still being researched, stemming from the plurality of human desiderata.

The relatively young age of XAI (and its discontinuity over time), also plays a part in this: some inconvenient truths are still under-researched.

#### 3.7.1 *Are explanations useful?*

A significant concern in XAI is that explanations are not actually practically useful to users, due to irrelevance, non-understandability or inaccuracy. This problem is exacerbated by the **lack of empirical human validation**: there is rarely concrete proof that these XAI methods have a practical benefit [257]. Lack of evaluation is its own issue, which is discussed in Section 3.8.

Some researchers argue that many explanation methods are presented as mathematical constructs that lack a practical justification [94]. Without a scenario or a problem being solved, these explanations have no use cases and no evaluation [252].

#### 3.7.2 *Plausible but not accurate, is that deceptive?*

A major pitfall of XAI is that explanations can be misleading by presenting information that appears plausible without being accurate or actually reflecting what is being explained.

Rudin *et al* discusses XAI methods for black box ML models, and claims that explanations for these are never true because they are inexact approximations [232]. Due to this, they hold the position that (at least for critical applications) the only honest method of obtaining explanations is by using a transparent model from the start. This paper is not entirely dismissive of XAI for black boxes, and proposes the replacement of the term “explanation” with “summary statistics”, and “trends”.

A serious issue is the possibility of **intentionally constructing deceptive explanations** via adversarial attacks, to which LIME and SHAP are vulnerable. For example, it is possible to build a classifier that contains a discriminatory bias and have it undetectable by LIME and SHAP, in more than one way [249, 80, 87]. Variable importance methods can be deceptive due to three main reasons:

- They are meant to be applicable to additive models without interactions [186], but these restrictions are clearly ignored, leading to an illusory understanding of how variables behave [112].
- There is no scientific evidence of their utility, and there is only anecdotal evidence.
- As discussed in [94] heat-maps and variable importance methods can say that “a variable was important in a prediction” but cannot say how exactly the hot spots are used by the model, which is an important factor [5].

In general, it is possible to manipulate explanations until they are palatable to users (but now inaccurate, perhaps intentionally), referred to as “explanation hacking” [258].

An issue which arises in most explanation methods is the “idealisation failure”, where a task is simplified in such a way to give digestible explanations, to the point of discarding the parts that are actually important [258].

### 3.7.3 *Transparent is not necessarily understandable*

While the structure of a model can be transparent in the sense that the individual parts are easy to understand, it might be the case that their composition leads to a system which cannot be understood [178].

This applies for example to overly deep decision trees: if a tree needs to be very deep in order to accurately capture the original function, its complete form cannot be understood, and its “simplified” form is likely omitting important details [23].

### 3.7.4 *Illusion of Understanding*

As discussed in [55], users are vulnerable to the **illusion of understanding**, which is a cognitive bias in which users (wrongly) believe they understand a system. Their self-reported confidence is initially high and then quickly falls when they have to put their understanding into practice.

This phenomenon contributes to an “easiness effect” where users uncritically accept simple justifications without deeper engagement [55]. To mitigate these issues, the authors of [55] propose the design of **XAI** methods that foster *reflection* rather than just passive acceptance of explanations. Therefore, interactive counterfactual explanations can be used to provoke deliberative thinking.

## 3.8 EVALUATION OF EXPLAINABILITY

Evaluation is a non-trivial aspect of explainability that is often overlooked.

**Only 0.7% of XAI papers contain empirical evidence of XAI usefulness through humans [257].** If we count evaluation where humans are not used, the percentage is still just 20% (as of 2020) [207].

The final aim of explanations is to serve a purpose for humans, but unfortunately human-based evaluation studies are time-consuming and expensive. Studies without human participants are more feasible, but unstandardised [182]. Therefore, explainability research is often accepted based on **anecdotal evidence** [207].

This contrasts with many of the principles of **XAI** - how explanations need to be at least **accurate** in order to be acceptable, and also **understandable** in order to be useful [192].

One of the main issues reported in the literature is the lack of standardised evaluation methods, especially due to the many desiderata of **XAI** [183]. In fact, it is not even clear if explainability should be evaluated qualitatively and/or quantitatively [83]. There are advantages and disadvantages for both:

- **Quantitatively:** proxy metrics such as explanation size (such as the depth of a tree) are sometimes used, but these are rather opaque and do not accurately capture human understanding.
- **Qualitatively:** interviews and observation studies can be used, but these can conceal undesired effects such as the illusion of understanding.

Some research argues that explanations should be measured in both **objective** and **subjective** metrics [62]. Objective metrics include **accurate interpretation** (the user can predict the consequences of their actions), and **accurate actionability** (the user can decide actions which lead to better results). Subjective metrics measure the user’s **trust** in their own choices, in the model (or metaheuristic in this case) and in the explanation

Based on how close the evaluation is to the real use cases, [83] suggests a taxonomy:

- **functionally grounded:** a proxy metric is used, and no humans are involved
- **application grounded:** the real users are used to evaluate the system, on the relevant use cases
- **human grounded:** these are scaled down version of application-grounded studies, where the original task has been simplified so that any lay-user can participate. Additionally, this reduces experiment time.

Although a true-scale study (with the real users and the real scenarios) would be the ideal evaluation method, real-world constraints force researchers to use proxy-evaluation methods.

For reviews on evaluation methods in **XAI**, we direct the reader to the work of Vilone and Longo [276, 275] for a detailed review of possible characterisations of explainability and evaluations, and to [207] for a more critical review of the literature for evaluation.

#### 3.8.0.1 *Quality metrics of explanations*

Within **XAI** literature there are many possible characterisations of what a “good” explanation looks like, with different works providing contradicting metrics [94]. The most commonly mentioned are **trust** [62] and quantifications of uncertainty [276], whereas for others the aim is improved **user performance** [252].

The latter is the most relevant to this thesis: the **presence of explanations should result in statistically provable improved user performance**, which can be quantitatively analysed. [62].

#### 3.8.1 *What makes evaluation so challenging?*

Evaluation is arguably one of the most challenging aspects of **XAI**, which can be attributed to the resources required to carry out human-based studies and the lack of commonly agreed-upon methods [182]. The ideal evaluation would involve many human domain experts interacting in real-world scenarios (an **application-grounded study**). These are unfeasible in terms of resources such as the availability of domain experts, especially for preliminary testing [83]. These participants may be hard to obtain, but their time is valuable and their quantity is scarce, both of which are not compatible with preliminary testing.

As a solution to this, the literature suggests **human-grounded** studies, where the original task is simplified to assess just human-usability of the explanations themselves [83]. The purpose of simplifying the task is to broaden the possible audience - to “lay-users”. These studies focus on whether the explanations are compatible with humans, so that a preliminary study can conclude whether they are worth exploring further. Because these studies use simplified problems, and they are tested on participants which differ from the final intended audience, these evaluations do not actually evaluate the original scenarios, and thus it can be hard to justify them (especially since they still require significant resources) [284].

#### 3.8.2 *Evaluation for methods similar to metaheuristics*

Despite the existence of various **XAI** for **metaheuristics methods**, there are few **studies which evaluate their explainability aspects**, and to the best knowledge of the authors they all relate to Human-In-The-Loop (**HITL**) approaches.

This could be caused by explanations for metaheuristics (excluding **HITL**) generally targeting researchers that are already familiar with metaheuristics rather than a general audience, such as Fitness Landscapes [267] and Local Optima Networks [209]. Due to the focus on researchers, there is less reason to conduct a wider user study.

**HITL** methods are notable also for having the most comprehensive user studies which evaluate the usability of their systems. The theses [177, 180] contain multiple user studies, but from the point of view of user interface design. A more relevant study is [280], where MAP-Elites is evaluated as a **HITL** tool to explore the search space. There are multiple works where the user is essentially an operator in the search process, such as [165, 204, 53, 47]. These works are of limited relevance due to the fact that while they are more transparent, they do not result in explanations and thus their evaluation methods are more aligned with User Experience rather than **XAI**. Nonetheless, these studies are worth noting due to the use of both objective and subjective metrics when evaluating the usability of the proposed tools.

This chapter will present the state of the art of methods which are relevant to our work, and discuss how our methods are novel.

The concept of **PSs** is both similar to existing methods, but also significantly different.

**PSs** also share a superficial resemblance with **schemata** (discussed in Section 4.1) and **backbones** (Section 4.2), but the underlying aim is different. **PSs** differ because they are instantiated and can be used for explainability. **PSs** can be seen as evolved indicators of quality, and in this sense they are similar to *innovization* principles, but they have completely different structures (discussed in Section 4.3).

**PSs** occupy the niche of **XAI for metaheuristics**, where many different and unrelated methods are present, such as Search Trajectory Networks (Section 4.4), Surrogate Model Mining (Section 4.6) and Search Trajectory Mining (Section 4.10). To our knowledge, similar methods do not exist for explaining optimisation problems. **PS** mining is available as an easily applicable addition to the tools available.

Within the **wider field of XAI** there are methods which are superficially similar, such as Learning Classifier Systems (Section 4.8) but with different aims. In terms of similarity in methodology, **PSs** are an example of explanations which are obtained *via* metaheuristics, and the most relevant work in **XAI** is anchors (Section 4.7).

As a summary of this chapter, **PSs** have different aims, different implementations and different results from the rest of the literature. It can be concluded that **PSs** are a complementary addition to **XAI** for metaheuristics.

#### 4.1 SCHEMATA

Schemata (discussed in Section 2.3.11 of Chapter 2) share structural similarities with **PSs**. In particular, **PSs** can be seen as an extension of schemata because both are patterns for combinatorial optimisation problems and are associated with sets of solutions. Moreover, the quality of both can be assessed using measures such as pattern size and mean fitness.

Despite these similarities, there are important distinctions which prevent direct comparisons. **Schemata are used for theoretical results**: they are derived analytically (i.e. by hand) for the analysis of algorithm convergence; **PSs** are mined and instantiated from empirical data. There are some studies which do generate patterns, but these are found for the purposes of improving algorithmic performance. For example, previous studies such as [91, 109, 254, 147] focus on algorithmic applications rather than human understandability.

Finally, **schemata do not explicitly account for linkage between variables**, which is why **PSs** incorporate the concept of *atomicity* to capture these interactions. While it would be misleading to claim that schemata and **PSs** are entirely separate, as **PSs** were strongly influenced by schemata, the differences in purpose, instantiation, and explainability justify treating them as distinct concepts.

#### 4.2 BACKBONES

In the context of satisfiability decision problems, satisfactory solutions often share a common pattern that is critical to ensuring their satisfiability. This pattern is referred to as the *backbone*, and its structure closely resembles that of schemata and **PSs** [250]. For a set of boolean logic statements where the satisfactory combinations of values are be [T F T T F], [F F T T F] and [T T F T F], the backbone is [\* \* \* T F].

Despite the structural resemblance, there are important distinctions in how backbones and **PSs** are used. **Backbones are specific to satisfiability problems**, and their formal definition focuses on the subset of variable assignments that guarantee satisfiability, rather than patterns associated with high fitness. Additionally, backbone research typically assumes the existence of a single backbone per problem, whereas **PSs** are designed to be multiple and potentially overlapping.

While some studies have extended backbone concepts to combinatorial optimisation problems (such as [175]), these works focus on specific metaheuristics and remain relatively limited compared to the broader research on **PSs**, which are applied to general metaheuristics.

#### 4.3 INNOVIZATION BY DEB

In the context of multi-objective optimisation, it has been observed that solutions on the Pareto front may share common patterns that are informative for understanding the underlying problem structure. Kalyanmoy Deb and collaborators introduced the concept of *innovization*, which seeks to identify such recurring patterns [77]. The central idea is that, by analysing the population of high-quality solutions, one can extract design principles or solution components that are consistently associated with superior performance.

Importantly, the various existing *innovization* papers (summarised in [104]) all acknowledge that there is **no established method to systematically extract these patterns**; instead their approach relies on metrics similar to simplicity to guide the search for relevant rules. The method implemented in [104] is the most relevant here – it finds mathematical invariants (equations which remain true) within the Pareto front. These rules are mathematical operations between continuous variables, such as  $x_2x_3 = 0.5$ , and they tend to follow strict formats [104].

The methodology developed in this thesis can be seen as an implementation of innovization, albeit much more general and applied to single-objective problems rather than multi-objective settings. The various papers relating to innovization apply ad-hoc methods for finding the rules for continuous problems, whereas we aim for a method which is generally applicable to combinatorial problems. Both approaches evolve their rules using a Multi-objective (**MO**) process, with the biggest difference being the objectives themselves - our objectives are defined over a  $P_{\text{Ref}}$  and include the concept of **atomicity**.

#### 4.4 SEARCH TRAJECTORY NETWORKS AND LOCAL OPTIMA NETWORKS

**Search Trajectory Networks** and **Local Optima Networks** are a popular way of analysing metaheuristic behaviour, and they differ from our methodology in what they explain.

Search trajectory networks represent how a metaheuristic “moves” between different regions of the search space, as a network graph [210]. Local Optima networks focus on local optima, and show how the metaheuristic can “transition” between them [267, 209].

These graphs reveal important characteristics of the search process. **Basins of attraction** are typically visible, as surrounding regions exhibit strong edges pointing toward them. If no outgoing edges are seen, it means that the metaheuristic may become trapped in local optima.

These methods explain **the metaheuristic**, and make assumptions about how the metaheuristic is implemented (for example Search Trajectory Networks require Iterated Local Search). The explanandum is the search process, whereas in our methodology it is the solutions themselves (either one at a time, or a collection of them).

#### 4.5 HUMAN IN THE LOOP APPROACHES

Human-In-The-Loop (**HITL**) methods integrate a human user into an algorithmic procedure. In the context of metaheuristics, this usually involves a user manually selecting preferred solutions at various stages of the search [166]. The human is directly involved in the search process, which inherently builds trust and understanding. A particularly noteworthy subset of **HITL** is Interactive Evolutionary Computation (**IEC**), in which the human acts as the fitness function [29]. This is especially useful in situations where the objective function is difficult or impossible to formalise, such as when subjective preferences are involved. By allowing explanations and outcomes to be tailored to the human evaluator, **IEC addresses the the personalisation of explanations**, and the tuning of the objective function through Inverse Reinforcement Learning [4].

**HITL** metaheuristics can be implemented in numerous ways, since any search operator can potentially be replaced by a human-in-the-loop decision. This makes these methods highly interdisciplinary, drawing on psychology, user interface design, and even art [166].

Despite their versatility and potential, HITL methods are often underrepresented in the literature of explainability.

When compared to our methodology, HITL approaches are essentially the inverse. In HITL, the user understands the algorithm because they actively participate in its execution, whereas our work focuses on conveying the end results of a metaheuristic to the user. Similarly, HITL relies on knowledge gained through experience in interacting with the algorithm, while our approach aggregates post-hoc information and presents it in a digestible format. Furthermore, HITL operates on a live, actively running algorithm and typically requires substantial modifications to its implementation, unlike our post-hoc analysis. These differences suggest that combining HITL approaches with our methodology could be highly fruitful.

#### 4.6 SURROGATE MODEL MINING

Surrogate models are an approach used to approximate the behaviour of a fitness function for an optimisation problem. They can be constructed either during the execution of a metaheuristic as part of an Estimation of Distribution Algorithm (EDA), or from a static dataset. In all cases, the surrogate model can provide information that is insightful and useful [278, 247].

For example, [278] uses a Markov model to predict solution fitness, where the model contains coefficients that quantify both variable importance and the linkage strength between variables. Similarly, other research has trained surrogate models during a Genetic Algorithm run and calculated variable importance by perturbing individual variables and observing the change in predicted fitness [173]. In this manner, one can estimate which variables and interactions are most relevant. In [247] Support Vector Machines (SVMs) were employed, whereas [278] explored linear regression, decision trees, and random forests. The main results of these works are variable importance metrics, which strongly differ from the aims of this thesis (discussed in in Section 4.9).

#### 4.7 ANCHORS

Anchors represent an interesting and under-represented tool in explainability research. Conceptually, anchors can be thought of as backbones for classification models: they identify the subset of the input that guarantees the predicted class remains unchanged [228]. An example is shown Figure 4.1. The general definition of anchors is intentionally broad, and they have been applied across tabular, continuous, combinatorial, image, and natural language data [283].

```
IF Country = United-States AND Capital Loss = Low
AND Race = White AND Relationship = Husband
AND Married AND 28 < Age ≤ 37
AND Sex = Male AND High School grad
AND Occupation = Blue-Collar
THEN PREDICT Salary > $50K
```

Figure 4.1: An example of an anchor, provided in [228]. A model was trained to determine whether a person would earn more or less than \$50'000 a year, and the rule shown guarantees a certain prediction.

The motivation behind anchors arises from a limitation in local explainability methods: lack of generality. Anchors aim to find rules that explain outputs for more general categories of inputs: their primary quality metric is *coverage*, which measures the extent to which an anchor reliably explains multiple inputs. These anchors are found via a heuristic, a self-assembly process[228].

There are several conceptual similarities between anchors and PSs, but they differ in how **anchors do not appear to consider the trade-off between multiple objectives (of understandability against coverage for example)**. This leads to different implementations: anchors are inferred from a stochastic self-assembly process, which differs from the multi-objective approach used for PS mining.

Both approaches are model- and problem-agnostic and can operate on a static dataset, but anchors are designed for classification tasks, whereas **PSs** are closer to regression since they relate to solutions associated with a scalar (the fitness value).

In summary, while anchors and **PSs** share a few high-level characteristics their similarities are limited. Anchors may be relevant in future work where **PS** mining is implemented differently.

#### 4.8 LEARNING CLASSIFIER SYSTEMS

Learning Classifier Systems (**LCSs**), in their classical formulation, are highly relevant to the study of explainable metaheuristics. They are systems designed to produce transparent models for classification and regression, where the underlying rules are essentially *schemata* [272]. During training, these rules are evolved in an **on-line learning process**: new rules are proposed and evaluated on incoming training instances, and their survival and reproduction depend on a fitness measure. This differs from standard evolutionary computation where the fitness of an individual is found immediately [272].

Learning Classifier System (**LCS**) are **PSs** are similar due to **convergent evolution** in research: both involve patterns defined for combinatorial problems, and these patterns are associated with measures of quality. Moreover, **LCSs** function as predictive models, analogous to the way **PSs** are used in decision trees. For example, XCSF is a popular **LCS** method where rules predict the mean value of the training instances they match, which aligns closely with the methodology used in our decision tree models for fitness estimation [288].

However, there are also significant differences. **LCSs** are primarily applied to classification and regression problems, so the quality metrics used to evaluate rules are typically accuracy and coverage of training samples. This does resemble our work on **PS4XRS** (discussed in [43] and Section 8.2), but not the general framework.

Additionally, **LCSs** implement heuristics to determine when one rule should replace another, rather than explicitly optimising a multi-objective search. Interpretability in **LCSs** is achieved indirectly, as simpler rules tend to dominate more complex ones, producing a transparent overall model. In contrast, **PSs** rely on explicitly defined quality metrics such as simplicity and atomicity to guide pattern selection and ranking, resulting in a more controlled and measurable explainability outcome.

Overall, **LCSs** provide a completely different approach to evolving rules for explanation, which could be combined with our methodology in future work.

#### 4.9 VARIABLE IMPORTANCE METHODS

Variable importance methods, such as **LIME** and **SHAP**, are intended to explain the contribution of each input variable to a **ML** model’s prediction. However, these approaches implicitly assume that variables contribute independently and additively to the outcome [112]. This assumption is often broken, especially where strong interaction effects between decision variables are common such as optimisation [59].

Consequently, their applicability to metaheuristics is limited: it is more useful as a measurement of certain properties relating to fitness (such as [139, 140]).

In contrast, the methodology presented in this thesis is designed to capture and represent these interactions through the use of patterns and linkage learning, offering insights that are more relevant for combinatorial optimisation problems. This is implemented by using the **atomicity** objective, as described.

#### 4.10 SEARCH TRAJECTORY MINING

*Fyvie et al’s* work also aims to improve the interpretability of metaheuristics, with a particular focus on the **Staff Rostering problem**. Central to his approach is the concept of trajectory mining, which was first introduced in [98]. The main idea is that the behaviour of a population-based metaheuristic can be understood by analysing the differences in genomes across generations. In this framework, each generation is summarised using descriptive statistics, allowing a “history” of the algorithm’s execution to be presented for explanatory purposes.

A key distinction from our work is that trajectory mining focuses primarily on the **behaviour of the algorithm** and its execution, rather than on the solutions themselves.

Another notable difference from our methodology is that Fyvie’s approach is applicable to population-based methods with ordered generations [99], whereas our work operates on unordered sets of solutions. Nonetheless, both approaches share the post-hoc nature and applicability to static datasets.

**MEASURING POPULATION DYNAMICS** The trajectory mining methodology was implemented in [102], where entropy and Principal Component Analysis (PCA) were used to characterise the generations of the algorithm. This allows researchers to compare different generations within a single run, compare different runs of the same algorithm, and even compare runs across different algorithms. Additionally, this approach enables the detection of phase shifts in the search process, distinguishing between exploration and exploitation phases.

**VARIABLE IMPORTANCE** The trajectory mining methodology was applied to the Staff Rostering problem in [100], producing generation-specific variable importance measures and introducing a novel trajectory mining method based on sensitivity analysis using linear models. The results of these novel methods were compared against existing approaches, showing consistency in the identified variable importance. It is worth noting that the Staff Rostering instance used in this work closely resembles the one discussed in [46], although it does not account for worker skills.

Later refinements of this work involved replacing PCA with multiple correspondence analysis, producing updated variable importance values that were compared against ANOVA. These analyses revealed, for instance, that workers with more rotas tended to have higher importance values. *Fyvie et al’s* methods were also applied to continuous optimisation problems in [102] and consolidated in [96].

**FINAL COMPARISON** When comparing Fyvie’s methodology to ours, it is evident that both approaches provide forms of summary statistics, though their goals differ. His work primarily aims to offer actionable insights on the behaviour of algorithms, whereas our approach focuses on actionable insights regarding solutions themselves. Furthermore, our method extends to non-population-based metaheuristics, while trajectory mining techniques are more restrictive. Despite these differences, it is sensible to consider the two methodologies as complementary, as both leverage static datasets to provide distinct and valuable explainability features.

#### 4.11 CONTRIBUTIONS AND NOVELTY IN CONTEXT

Having discussed relevant works in the literature, a summary of the contributions and novelty of our research work is provided.

Firstly, the concept of **PSs** is built on the idea that **explanations are the result of a multi-objective optimisation problem**. This is discussed in the literature ([14]) and implemented indirectly by anchors (Section 4.7) and **LCSs** (Section 4.8), whereas we actually implement it in full. This is implemented for innovation, but its use is limited by the objectives not being explored beyond mean-fitness. **PSs** are a concrete and explicit resolution of an optimisation task, unlike schemata which are a theoretical construct.

Secondly, the **explanation methods proposed in this thesis are designed for optimisation tasks** as the main use-case scenario, rather than as an afterthought. As discussed in Section 3.1, **XAI** is dominated by tools for **ML**, and explanations for optimisation tend to be an afterthought. Methods such as anchors, Learning Classifier Systems and variable importance methods (discussed in Section 4.9) can be applied to optimisation, but by doing so many assumptions and domain-specific requirements are ignored. For example, variable importance methods tend to ignore interaction effects, which we argue are a core characteristic of complex optimisation problems, and our linkage learning research is designed to detect such effects.

Beyond these major contributions, there are smaller but essential contributions which differ from the literature:

- The concept of **atomicity** as an objective does exist in the literature of **EDAs** but is never used for explainability purposes and acts as a proxy for other purposes. In our work it is developed and expanded upon (for example via Independence, Section 6.8).

- The other objectives and the attached discussions (Section 6.7, Section 6.9, Section 6.8 and Section 6.10) illustrate the novel idea that there is no universally satisfactory set of objectives for explanations, and rather that the responsibility of the system is to allow the user to choose the objectives which best suit their motivations.
- The production of a Pareto Front of explanations, rather than forcing the user into a specific trade-off between explanatory objectives. This is discussed in the theory of XAI but it is not practically seen in the literature.
- The use of descriptors (Section 6.12) as a tool to present to the user possibly useful associations between values.
- The creation of decision trees based on PSs, as discussed in Section 8.3.2. These can introduce linkage information into their decision nodes, as to better reflect the real structure of the optimisation problem.
- The explanations produced in the PS4XRS project are novel, since no similar explanations are present in the literature of Recommender Systems.
- The user study discussed in Chapter 9 is novel in how it is an evaluation of explanations for optimisation scenarios (not seen in the literature) that is also practically and quantitatively based (rarely seen in XAI, with the closest example being [184]).



### Part III

## METHODOLOGY: PARTIAL SOLUTIONS

# 5

## INTRODUCTION TO PARTIAL SOLUTIONS

### 5.1 BRIEF INTRODUCTION

For a combinatorial search space of  $L$  variables, solutions are vectors of symbolic values (although these symbols are often numbers for convenience). An example of a solution would be (5 1 2 0 4).

Consider a combinatorial search space of  $L$  variables, where each variable (indexed by  $i$ ) can take one value from a finite set  $D_i$ . In this search space, a solution can be written as:

$$s = (s_1, s_2, \dots, s_L), \text{ where } s_i \in D_i \quad (5.1)$$

A **Partial Solution** has the same structure as a solution, but any variable can also take on the "\*" wildcard value.

An example **PS** would be (\* \* 2 0 \*). For convenience, a **PS** will be indicated with lowercase letter with apex  $\psi$ , such as  $s^\psi$ .

For the search space of  $L$  variables where each variable can have values  $D_i$ , a Partial Solution can be written as:

$$s^\psi = (s_1^\psi, s_2^\psi, \dots, s_L^\psi) \text{ where } s_i^\psi \in [D_i \cup \{*\}] \quad (5.2)$$

These **Partial Solutions** describe sub-configurations of values observed in *desirable* solutions, which usually means that they have high fitness.

**PSs** are patterns intended for explainability, and they are generated explicitly - they are found by using a search process in which the objectives are based on how easy these patterns are to understand, or how often they are associated with high fitness. The combination of objectives chosen controls the nature of the patterns *generated*. This process, **PS mining**, is implemented as a Multi-objective (**MO**) search and is described in Chapter 6. The current chapter will instead focus on the core definitions and theoretical aspects of **PS**.

The core ideas as shown in Figure 5.1: given a dataset of evaluated solution, there are many possible **PSs**, but are interested in finding the ones which will serve best as explanations.

This chapter concludes with an example of how **PSs** may be applied on an optimisation problem, in Section 5.7.

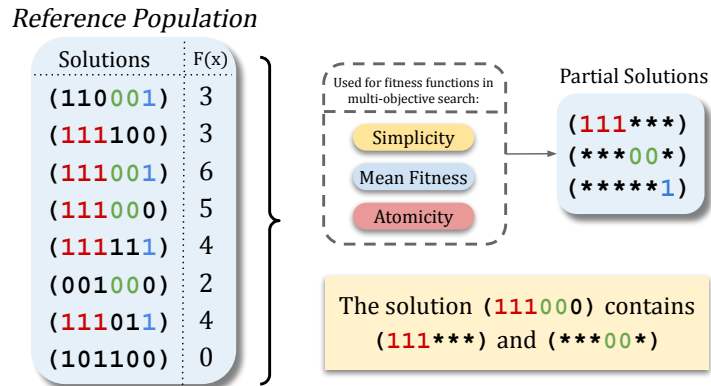


Figure 5.1: Given a population of solution, we find **PS** that can describe it well

## 5.2 SUMMARY OF PAPERS

The papers produced as part of this research are summarised here. **PSs** were first introduced in [44], and then applied in [46] and in [43]. Having defined the method which finds useful **PSs**, the patterns can be used in multiple ways beyond just explanations. In [44] they are used to construct solutions, and in [45] they are used to predict solution fitness. Finally, a user study was designed to test the effectiveness of **PSs** in [42] (specifically the models produced in [45]). Chronologically, the publications produced were:

- [44]: **An introduction to PSs**. (Attached in Chapter A). This paper introduced the concept of **PSs**, and the main objectives used throughout this thesis: **\*-count**, **simplicity**, **mean-fitness** and **atomicity**. The method used to find them is closer to a single-objective Iterated Local Search.
- [46]: **Application of PSs to the SR problem**. Here **PSs** are found via a multi-objective metaheuristic, which will remain the method in use hereafter. The methodology is applied on the Staff Rostering problem, demonstrating the explanations obtained.
- [45]: **Making decision trees based on PSs**. **PSs** are used to create transparent models which can predict the fitness of solutions. The models are competitive with traditional decision trees.
- [42]: **Design of a user study for explanations**. This paper revolves around the design and deployment of a user study in order to determine if and how much our explanations are beneficial to **BT**'s use case scenarios.
- [43]: **PSs are used to explain the performance of a machine learning model in a Recommender system**. **PSs** are adapted to work on a dataset which is not derived from a metaheuristic. The core methodology is maintained, and it is shown how explanations produced are usable on unseen data.

## 5.3 ORIGIN

The name **Partial Solution** originates from the Optimal Mixing literature, in particular from the abstract of “Optimal Mixing Evolutionary Algorithms” by *Thierens et al* ([266]):

A key property of GAs and EDAs is their ability to juxtapose **Partial Solutions** or substructures from different solutions to form new good solutions.

Optimal Mixing, as well as DSGMA-II [266, 136] were the inspiration for a system that would *explicitly* search patterns that are common in high-fitness solutions.

## 5.4 FORMAL DEFINITION

Here we formally define **PSs**, so that the operations described later are unambiguous.

### 5.4.1 Properties of Partial Solutions

When a variable has a non-\* value, it is called a **fixed variable**.  $\text{fixed\_count}(x^\psi)$  indicates how many variables have a non-\* value within  $x^\psi$ .

$$\text{fixed\_count}(x^\psi) = |\{i \in [1..L-1] \text{ s.t. } x_i^\psi \neq *\}| \quad (5.3)$$

This attribute is not controlled directly, and instead it is intentionally left to the **PS** mining process to find patterns with different quantities of fixed variables.

The most important part about **PSs** is that they are contained within solutions - we say a  $x^\psi$  **matches** a solution  $s$  if the pattern is *present* in the solution (defined in 5.4).

A pattern may be **applied** on a solution, so that a new solution is obtained (defined in 5.5).

$$\text{match}(x^\psi, s) = \forall i \in [1..L], x_i^\psi = s_i \vee x_i^\psi = * \quad (5.4)$$

$$\text{apply}(x^\psi, s) = t, \quad t_i = \begin{cases} s_i & x_i^\psi = * \\ x_i^\psi & \text{otherwise} \end{cases} \quad (5.5)$$

The process which finds useful patterns uses as input a collection of evaluated solutions, the **Reference population**  $P_{\text{Ref}}$ . This is an unordered set of solutions, which have been evaluated as part of a metaheuristic process. In practical terms, this data is a byproduct of a search process, which is then used to apply our methodology.

The  $P_{\text{Ref}}$  can be partitioned into solutions which contain a certain **PS**, and solutions which do not. These are called  $P_{\text{match}}$  and  $P_{\text{notmatch}}$  respectively.

$$P_{\text{match}} = \{s \in P_{\text{Ref}} \mid \text{match}(x^\psi, s)\} \quad (5.6)$$

$$P_{\text{notmatch}} = P_{\text{Ref}} \setminus P_{\text{match}} \quad (5.7)$$

## 5.5 SEARCH SPACES FOR PARTIAL SOLUTIONS

Although normal solutions have a certain search space, **PSs** have their own extended search space. This is particularly relevant when implementing **PS** mining, as there is a choice of what search space to consider:

- **Subset-of-Solution**: the aim is to find the patterns which explain a specific solution, and we only consider patterns which are contained within it. If the solution to explain is (1 2 3 4), then possible results are (1 2 \* \*), (\* \* 3 4), or (1 2 3 \*).
- **Any-Pattern**: there is no restriction to what pattern we can accept, as long as it has the right format and all fixed values are valid.

These two search spaces correspond to the two major kinds of explanations: **global** and **local**. The **Subset-of-Solution** is suited for those who want to understand a specific solution, whereas **Any-Pattern** is used to understand the  $P_{\text{Ref}}$  and find any “loopholes” in the fitness function.

## 5.6 POST-PROCESSING OF PARTIAL SOLUTIONS

**PSs** by themselves can be hard to understand, so this thesis will also propose post-processing steps to make them more palatable as explanations.

The **PS** mining process produces many **PSs**, and post-processing can filter them and provide likely causes for their existence.

The post-processing steps can be summarised as:

1. **Optionally cluster** all the **PSs**, based on the methodology described in Section 6.13. For real problems, the patterns produced have a tendency to come in “families” with only minor variations between their members. By clustering, the user is shown a representative example from each family. The patterns (1 2 \* 0 1 1), (1 2 \* 0 \* 1), and (1 2 \* 0 \* \*) would be clustered together and represented by (1 2 \* 0 \* 1).
2. For each pattern to be shown, give information relating to its **effect on fitness**. This includes descriptive statistics based on the fitness values of the solutions matched by the pattern, such as average, standard deviation *etc.* A non-parametric test (as defined in Section 6.7) can be applied to verify if the fitness of matched solutions is greater (or different, depending on the task) from the fitness values of non-matching solutions, and the p-value can be presented to the user.
3. The fitness function may be decomposed into smaller functions, which we call **proxy-fitness functions**. If these are provided, our system will automatically

check if a **PS** is associated with high or low values for these proxy fitness functions. If the patterns associated with high fitness are also associated with relatively high or low values for certain proxy fitness functions, this information can help guide the user in determining how that proxy-fitness affects overall solution fitness. Proxy-fitness functions which take relatively high or low values are given the name of **descriptors**, and the algorithm which finds them is shown in Section 5.6.

## 5.7 A PRACTICAL EXAMPLE

This section aims to give a more practical understanding of what the typical process of using our methodology looks like. For this section, we introduce an optimisation problem which is an instance of a  $(0, n)$  **Knapsack** problem.

### 5.7.1 The problem

It might seem counter-productive to introduce a new problem just for this example. Although the Staff Rostering problem could have been used, it is too complex as a starting example. Explanations for the Staff Rostering problem are presented in [46].

Moreover, benchmark problems such as RoyalRoad are not suitable due to their abstract nature: there is no meaning to their variables, and thus these problems lack a layer of realism. Therefore, we focus on a well known problem: **Knapsack**.

Knapsack problems relate to a variety of other optimisation problems, which have been shown to be complex for both computers and humans, and are encountered in many daily scenarios [203].

We present here a fairly mundane task: buying grocery items with the aim of maximising the quantity of calories. The scenario is the following:

You have a list of 30 possible food items (Table 5.1), and for each you can decide to buy 0, 1 or 2 items. The aim is to **maximise the quantity of calories**. The constraints are price, volume and weight: you are allowed to spend £50, the items have to fit into two bags-for-life, and together the bags should not weigh more than 15kg.

This problem is not the typical 0-1 Knapsack problem, which is to underline the fact that our methodology works for any cardinality of variables rather than just binary.

Table 5.1 was obtained via cursory research of UK stores. The exact figures are not too important: what matters is that the reader will have some intuition and expectation for how to solve the problem, and we intend to show how the proposed methodology works alongside human intuition.

The fitness function for such a problem is the sum of calories of the items purchased, but it is 0 if any of the constraints are violated, as defined in Equation (5.12).

Let:

- **calories(i)** be the calories of the  $i$ th item,
- **cost(i)** be the cost of the  $i$ th item (in pence),
- **weight(i)** be the weight of the  $i$ th item (in grams),
- **volume(i)** be the space occupied by the  $i$ th item, as a percentage of a bag-for-life,
- **cost\_budget** represents the £50 budget, which will be set to 5000,
- **weight\_budget** represents the 15kg budget, which will be set to 15000,
- **volume\_budget** represents the 2 bag budget, and it is set to 200

From these, the fitness function  $F$  is defined:

$$\text{total\_calories}(x) = \sum_{i=0}^L \text{calories}(i) \cdot x_i \quad (5.8)$$

$$\text{total\_cost}(x) = \sum_{i=0}^L \text{cost}(i) \cdot x_i \quad (5.9)$$

$$\text{total\_weight}(x) = \sum_{i=0}^L \text{weight}(i) \cdot x_i \quad (5.10)$$

$$\text{total\_volume}(x) = \sum_{i=0}^L \text{volume}(i) \cdot x_i \quad (5.11)$$

$$F(x) = \begin{cases} 0 & \text{if } \text{total\_cost}(x) > \text{cost\_budget} \\ 0 & \text{if } \text{total\_weight}(x) > \text{weight\_budget} \\ 0 & \text{if } \text{total\_volume}(x) > \text{volume\_budget} \\ \text{total\_calories}(x) & \text{otherwise} \end{cases} \quad (5.12)$$

This formulation of a constraint, where it is directly part of the fitness function is termed a “penalty function”. This allows a simpler formulation (hence it was used here to keep the example shorter) but it also negatively affects the search process [57]. The specific approach used here is termed “death penalty”, and should not be used in real-world cases, and instead the constraints should be encoded separately [57]. It was used here since solving the original problem is not the topic of discussion, and allows a much more concise problem formulation.

As well as a fitness function, there are several proxy-fitness functions are relevant to  $F$ :

- **total\_calories, total\_cost, total\_weight, total\_volume**: as defined above
- **calorie/cost ratio, calorie/weight ratio, cost/volume ratio** etc...: various ratios between the totals obtained. The concept of value “density” (defined in [152]) arises quickly when reasoning about this problem.
- **cost\_constraint, weight\_constraint, volume\_constraint**: binary values representing whether a solution satisfied each constraint.

### 5.7.2 Solving the problem and generating the dataset

For this example, a typical Genetic Algorithm with  $\mu + \lambda$  strategy is used to solve the optimisation problem. The metaheuristic algorithm is not relevant to this example, what is important is how it needs to be modified in order to apply our methodology.

Whenever a solution is evaluated, it also needs to be added to a data structure alongside its fitness (and any relevant proxy-fitness values). At the end of the metaheuristic process, the solutions are stored as a `.csv` file, which will be our  $P_{\text{Ref}}$ . These modifications are minimal, and it is likely that the changes in code are not within the implementation of the metaheuristic itself.

### 5.7.3 Methodology: loading the dataset

This example is produced by using our methodology. Outputs produced by the current implementation are presented in the **yellow** boxes.

The user first needs to select the  $P_{\text{Ref}}$  data, and the system can show how many unique solutions were found, and the optima.

Table 5.1: Table showing the name, calories, cost, weight and volume of each purchasable item in the example optimisation problem. The volume refers to the percentage of space which an item would occupy in a bag-for-life.

| name                          | calories | cost | weight | volume |
|-------------------------------|----------|------|--------|--------|
| Apples (Pack of 6)            | 312      | 220  | 900    | 15%    |
| Banana (Single)               | 105      | 18   | 120    | 3%     |
| Sourdough Loaf                | 1850     | 180  | 800    | 22%    |
| Semi-Skimmed Milk (2 Pints)   | 568      | 125  | 1136   | 15%    |
| Cheddar Cheese (400g)         | 1664     | 350  | 400    | 4%     |
| Free Range Eggs (Box of 6)    | 408      | 195  | 420    | 9%     |
| Chicken Breast Fillets (300g) | 318      | 420  | 300    | 6%     |
| Minced Beef 12% Fat (500g)    | 1125     | 349  | 500    | 6%     |
| Bacon (Pack of 10)            | 980      | 275  | 300    | 6%     |
| Vegan Sausages (Pack of 6)    | 790      | 250  | 300    | 7%     |
| Carrots (1kg Bag)             | 410      | 65   | 1000   | 9%     |
| Broccoli Head                 | 125      | 75   | 350    | 10%    |
| Baby Spinach (250g)           | 58       | 150  | 250    | 5%     |
| Red Potatoes (2.5kg)          | 1925     | 199  | 2500   | 20%    |
| Avocado (Ripe)                | 240      | 95   | 200    | 4%     |
| Orange Juice Smooth (1L)      | 430      | 185  | 1000   | 12%    |
| Porridge Oats(1kg)            | 3710     | 140  | 1000   | 15%    |
| Heinz Baked Beans (415g Tin)  | 324      | 140  | 415    | 5%     |
| Basmati Rice (1kg)            | 3500     | 180  | 1000   | 7%     |
| Pasta Fusilli (500g)          | 1785     | 75   | 500    | 6%     |
| Pasta Sauce (500g)            | 220      | 160  | 500    | 6%     |
| Milk Chocolate Bar (100g)     | 53       | 135  | 100    | 2%     |
| Digestive Biscuits (400g)     | 1940     | 85   | 400    | 6%     |
| Crisps (150g)                 | 765      | 199  | 150    | 25%    |
| Greek Yogurt (500g)           | 480      | 110  | 500    | 7%     |
| Frozen Garden Peas (1kg)      | 690      | 145  | 1000   | 8%     |
| Frozen Pizza (Pepperoni)      | 840      | 300  | 450    | 25%    |
| Diet Coke (2L)                | 8        | 210  | 2000   | 15%    |
| Instant Coffee (100g Jar)     | 10       | 350  | 100    | 5%     |
| Tea Bags (Pack of 50)         | 0        | 220  | 100    | 5%     |

Loaded “pref.csv”, which contains 8703 solutions. The optima was [...], with fitness [...].

Then, a linkage table is generated. Linkage as a concept will be explained in Chapter 7, but in summary it indicates how much one variable’s effect on fitness is influenced by other variables. Presence of linkage is important in order to know if our methodology will be useful or not.

In this case, the user will be using dominance-based linkage by Cramer’s V, shown in Figure 5.2).

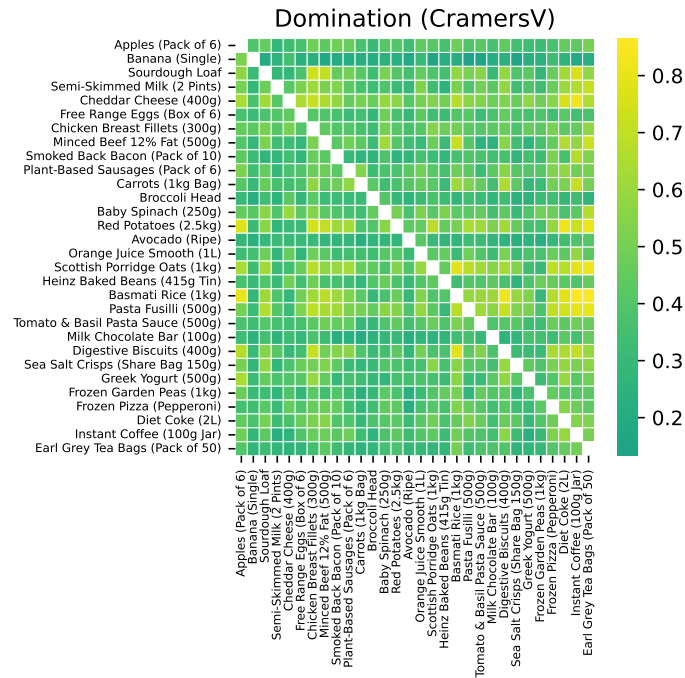


Figure 5.2: Plot showing the detected linkage strength through “Domination-based” linkage learning. The cell at [row]-[column] has a high value (lighter colour) if the item <row> has an effect on fitness dependent on <column>.

From this table, we see that carb-heavy items (Pasta, Oats, Rice, Potatoes) interact with some non-caloric items (apples, Diet coke, Instant coffee, etc..). These relationships can be attributed to constraint satisfaction (e.g. potatoes are good because of their calories, but since they are big then they might break the volume constraint if bought with apples as well).

The presence of such linkage is also a sign that variable importance is not “the full story”, and that PSs will be a useful explanation method.

#### 5.7.4 Finding Partial Solutions

Here, the user can experiment with various possible search metrics in order to find useful patterns. For brevity, here we use **\*-count**, **simplicity**, **mean-fitness** and **atomicity**.

This produces a Pareto front consisting of many similar patterns. To reduce the quantity of information to the user, only a representative subset of the Pareto front is shown. This is implemented by first clustering all the patterns within the Pareto Front, and then extracting the centroid of each cluster, as described in Section 6.13.

The PSs which will be shown as a **demonstration** of what the outputs of our methodology are. They are not to be taken as an evaluation, and thus an investigation of whether these explanations are “correct” is beyond the scope of this demonstration. They are presented here as they would be to a would-be user.

#### 5.7.4.1 Local Explanations

The single-objective optimisation Knapsack problem was formulated and then passed to the GA. The highest fitness solution found is the following:

| ITEMS |                               |   |                              |
|-------|-------------------------------|---|------------------------------|
| 0     | Apples (Pack of 6)            | 0 | Orange Juice Smooth (1L)     |
| 0     | Banana (Single)               | 2 | Porridge Oats(1kg)           |
| 2     | Sourdough Loaf                | 2 | Heinz Baked Beans (415g Tin) |
| 0     | Semi-Skimmed Milk (2 Pints)   | 2 | Basmati Rice (1kg)           |
| 2     | Cheddar Cheese (400g)         | 2 | Pasta Fusilli (500g)         |
| 0     | Free Range Eggs (Box of 6)    | 0 | Pasta Sauce (500g)           |
| 0     | Chicken Breast Fillets (300g) | 0 | Milk Chocolate Bar (100g)    |
| 2     | Minced Beef 12% Fat (500g)    | 2 | Digestive Biscuits (400g)    |
| 2     | Bacon (Pack of 10)            | 0 | Crisps (150g)                |
| 2     | Vegan Sausages (Pack of 6)    | 1 | Greek Yogurt (500g)          |
| 0     | Carrots (1kg Bag)             | 2 | Frozen Garden Peas (1kg)     |
| 0     | Broccoli Head                 | 0 | Frozen Pizza (Pepperoni)     |
| 0     | Baby Spinach (250g)           | 0 | Diet Coke (2L)               |
| 0     | Red Potatoes (2.5kg)          | 0 | Instant Coffee (100g Jar)    |
| 2     | Avocado (Ripe)                | 0 | Tea Bags (Pack of 50)        |

A search of the PSs that explain this solution results in 100 patterns. After clustering, 3 are presented.

#### FIRST LOCAL PARTIAL SOLUTION

| PATTERN                                                   |                             |                              |                               |
|-----------------------------------------------------------|-----------------------------|------------------------------|-------------------------------|
| 0                                                         | Semi-Skimmed Milk (2 Pints) | 0                            | Chicken Breast Fillets (300g) |
| 0                                                         | Frozen Pizza (Pepperoni)    |                              |                               |
| Mean fitness is in the top 44%, p-value for test: < 0.001 |                             |                              |                               |
| DESCRIPTORS                                               |                             |                              |                               |
| Descriptor                                                | Avg                         | Position                     |                               |
| cal/volume                                                | 180.68                      | (HIGHER than normal, top 2%) |                               |
| cal/cost                                                  | 7.815                       | (HIGHER than normal, top 2%) |                               |
| cal/weight                                                | 2.575                       | (HIGHER than normal, top 4%) |                               |

The pattern above is in the negative, it represents cases where Milk, Pizza and Chicken Breasts are not purchased. In the trade-off between simplicity and effect on fitness, this pattern leans more towards simplicity: the mean fitness of its solutions is in the top 44%, which is not very high. Nonetheless, its positive effect is attested by the p-value of a statistical test. There are 3 descriptors provided for this pattern, which suggest that the ratio of calories is high relative to volume, weight and cost. This is slightly confusing for a “negative” pattern, but it means that by *avoiding* certain items we can keep the calorie-density high.

## SECOND LOCAL PARTIAL SOLUTION

| PATTERN                                                             |                               |                              |                             |
|---------------------------------------------------------------------|-------------------------------|------------------------------|-----------------------------|
| 0                                                                   | Apples (Pack of 6)            | 0                            | Semi-Skimmed Milk (2 Pints) |
| 0                                                                   | Chicken Breast Fillets (300g) | 0                            | Red Potatoes (2.5kg)        |
| 0                                                                   | Orange Juice Smooth (1L)      | 2                            | Porridge Oats(1kg)          |
| 0                                                                   | Frozen Pizza (Pepperoni)      | 0                            | Diet Coke (2L)              |
| <i>Mean fitness is in the top 43%, p-value for test: &lt; 0.001</i> |                               |                              |                             |
| DESCRIPTORS                                                         |                               |                              |                             |
| Descriptor                                                          | Avg                           | Position                     |                             |
| cal/volume                                                          | 184.15                        | (HIGHER than normal, top 6%) |                             |

This pattern is from a different cluster, and it is more complex. Again, it is mostly in the negative but it also includes the purchase of 1Kg of oats. There is only one descriptor in this case, so this pattern is not as easy to understand as the first. It appears that the pattern is associated with higher caloric density with respect to volume.

## THIRD LOCAL PARTIAL SOLUTION

| PATTERN                                                             |                            |                              |                               |
|---------------------------------------------------------------------|----------------------------|------------------------------|-------------------------------|
| 0                                                                   | Apples (Pack of 6)         | 0                            | Semi-Skimmed Milk (2 Pints)   |
| 0                                                                   | Free Range Eggs (Box of 6) | 0                            | Chicken Breast Fillets (300g) |
| 0                                                                   | Carrots (1kg Bag)          | 0                            | Broccoli Head                 |
| 0                                                                   | Red Potatoes (2.5kg)       | 0                            | Orange Juice Smooth (1L)      |
| 2                                                                   | Porridge Oats(1kg)         | 2                            | Basmati Rice (1kg)            |
| 0                                                                   | Sea Salt Crisps            | 0                            | Frozen Pizza (Pepperoni)      |
| 0                                                                   | Diet Coke (2L)             | 0                            | Instant Coffee (100g Jar)     |
| 0                                                                   | Tea Bags (Pack of 50)      |                              |                               |
| <i>Mean fitness is in the top 37%, p-value for test: &lt; 0.001</i> |                            |                              |                               |
| DESCRIPTORS                                                         |                            |                              |                               |
| Descriptor                                                          | Avg                        | Position                     |                               |
| cost_constraint                                                     | 0.979                      | (HIGHER than normal, top 1%) |                               |
| weight_constraint                                                   | 0.990                      | (HIGHER than normal, top 1%) |                               |
| volume_constraint                                                   | 0.819                      | (HIGHER than normal, top 1%) |                               |
| calories                                                            | 35876                      | (HIGHER than normal, top 1%) |                               |
| cal/cost                                                            | 8.114                      | (HIGHER than normal, top 1%) |                               |
| cal/weight                                                          | 2.700                      | (HIGHER than normal, top 1%) |                               |
| cal/volume                                                          | 190.14                     | (HIGHER than normal, top 1%) |                               |
| cost/weight                                                         | 0.333                      | (HIGHER than normal, top 1%) |                               |
| cost/volume                                                         | 23.482                     | (HIGHER than normal, top 1%) |                               |

This pattern is from another cluster, and it is yet again mostly in the negative. It is a rather complex pattern, but many descriptors are provided. The most interesting descriptors are the first three, indicating that the pattern is associated with solutions that satisfy the three constraints, which is necessary in order to have high fitness.

As well as this, calories are high, and various ratios relating to calories are also high, which indicates that this pattern is highly exploitative of fitness.

Additionally, it also notes that the ratios of cost/weight and cost/volume are high, which is unlikely to be helpful. In this case, the descriptors might not actually be useful, and in general it shows that descriptors show correlation but not necessarily causation.

#### 5.7.4.2 Global Explanations

Here we search for any useful patterns for this task, based on the given reference population. The search uses the same objectives as the previous section, but a different search space (Any-Pattern) and different operators (as described in Section 6.5.2). The result is 54 patterns, which are divided into 4 groups after clustering. For each cluster, we present the centroid.

##### FIRST GLOBAL PARTIAL SOLUTION

| PATTERN                                                             |                               |                                |                             |
|---------------------------------------------------------------------|-------------------------------|--------------------------------|-----------------------------|
| 0                                                                   | Sourdough Loaf                | 0                              | Semi-Skimmed Milk (2 Pints) |
| 0                                                                   | Chicken Breast Fillets (300g) | 0                              | Frozen Pizza (Pepperoni)    |
| <i>Mean fitness is in the top 34%, p-value for test: &lt; 0.001</i> |                               |                                |                             |
| DESCRIPTORS                                                         |                               |                                |                             |
| Descriptor                                                          | Avg                           | Position                       |                             |
| cal/volume                                                          | 204.50                        | (HIGHER than normal, top 1%)   |                             |
| cost/volume                                                         | 27.003                        | (HIGHER than normal, top 2%)   |                             |
| volume                                                              | 159.88                        | (LOWER than normal, bottom 2%) |                             |
| weight/volume                                                       | 78.390                        | (HIGHER than normal, top 3%)   |                             |
| volume_constraint                                                   | 0.964                         | (HIGHER than normal, top 3%)   |                             |
| cal/weight                                                          | 2.603                         | (HIGHER than normal, top 3%)   |                             |
| weight                                                              | 12481                         | (LOWER than normal, bottom 6%) |                             |

This pattern is yet again in the negative. It appears that for this problem, the main priority in order to have good fitness lies in NOT bringing items that are likely to break constraints. This specific pattern appears to have high density over volume, and indeed the volume is lower than normal. Looking at the items, this makes sense: we avoid a loaf of bread, chicken breast fillets, milk and pizza. Note that this pattern is almost identical to what was found in the local explanations, with the difference being the loaf of bread. This pattern does not match the solution found before.

##### SECOND GLOBAL PARTIAL SOLUTION

| PATTERN                                                             |                            |                                |               |
|---------------------------------------------------------------------|----------------------------|--------------------------------|---------------|
| 0                                                                   | Vegan Sausages (Pack of 6) | 0                              | Broccoli Head |
| <i>Mean fitness is in the top 41%, p-value for test: &lt; 0.001</i> |                            |                                |               |
| DESCRIPTORS                                                         |                            |                                |               |
| Descriptor                                                          | Avg                        | Position                       |               |
| volume_constraint                                                   | 0.785                      | (HIGHER than normal, top 2%)   |               |
| cost_constraint                                                     | 0.919                      | (HIGHER than normal, top 3%)   |               |
| volume                                                              | 192.46                     | (LOWER than normal, bottom 3%) |               |
| cost                                                                | 4330.9                     | (LOWER than normal, bottom 3%) |               |
| weight_constraint                                                   | 0.861                      | (HIGHER than normal, top 5%)   |               |

This pattern is simple, with only two items, and yet it appears to have a detectable effect on fitness. This seems to be achieved by constraint satisfaction, based on the descriptors provided. In particular, avoiding vegan sausages and broccoli helps keep the volume and cost down.

##### THIRD GLOBAL PARTIAL SOLUTION

| PATTERN                                                             |                |                                |                            |
|---------------------------------------------------------------------|----------------|--------------------------------|----------------------------|
| 2                                                                   | Sourdough Loaf | 0                              | Vegan Sausages (Pack of 6) |
| <i>Mean fitness is in the top 41%, p-value for test: &lt; 0.001</i> |                |                                |                            |
| DESCRIPTORS                                                         |                |                                |                            |
| Descriptor                                                          | Avg            | Position                       |                            |
| cal/cost                                                            | 7.763          | (HIGHER than normal, top 2%)   |                            |
| cost_constraint                                                     | 0.919          | (HIGHER than normal, top 3%)   |                            |
| volume_constraint                                                   | 0.730          | (HIGHER than normal, top 3%)   |                            |
| cost                                                                | 4328.4         | (LOWER than normal, bottom 3%) |                            |

This pattern is also simple, but has a less strong effect on fitness. The pattern suggests that by buying two loaves of bread and avoiding sausages, one can improve the calorie/cost density by lowering the cost. Moreover, solutions containing this pattern are more likely to satisfy cost and volume constraints. This pattern resembles the one previously.

#### FOURTH GLOBAL PARTIAL SOLUTION

| PATTERN                                                             |                            |                                |                            |
|---------------------------------------------------------------------|----------------------------|--------------------------------|----------------------------|
| 2                                                                   | Minced Beef 12% Fat (500g) | 0                              | Vegan Sausages (Pack of 6) |
| 1                                                                   | Carrots (1kg Bag)          | 0                              | Broccoli Head              |
| <i>Mean fitness is in the top 16%, p-value for test: &lt; 0.001</i> |                            |                                |                            |
| DESCRIPTORS                                                         |                            |                                |                            |
| Descriptor                                                          | Avg                        | Position                       |                            |
| cal/cost                                                            | 8.649                      | (HIGHER than normal, top 2%)   |                            |
| cal/volume                                                          | 190.13                     | (HIGHER than normal, top 2%)   |                            |
| volume_constraint                                                   | 1.000                      | (HIGHER than normal, top 2%)   |                            |
| weight_constraint                                                   | 1.000                      | (HIGHER than normal, top 4%)   |                            |
| cost                                                                | 4085.0                     | (LOWER than normal, bottom 4%) |                            |
| volume                                                              | 185.44                     | (LOWER than normal, bottom 5%) |                            |
| cost_constraint                                                     | 1.000                      | (HIGHER than normal, top 5%)   |                            |
| weight/volume                                                       | 74.822                     | (HIGHER than normal, top 5%)   |                            |

This pattern is a more specialised version of the previous explanation. This pattern has a very strong effect on fitness (top 16% mean fitness), which appears to be by having low cost and low volume. In turn, this causes high calorie density and means that solutions containing this pattern are more likely to satisfy constraints.

## 6.1 INTRODUCTION

Desirable **PSs** are inferred from a set of solutions, and this chapter will describe how this task can be achieved by formulating it as a multi-objective search problem.

After deciding the **PS** search space, and the objectives (typically simplicity, mean fitness and atomicity), the Multi-objective (**MO**) problem is solved by a metaheuristic and a Pareto front of **PSs** is obtained.

This approach allows for the objectives to be easily changed and customised, as well as allowing the user to select which trade-off between the objectives they prefer.

This procedure, **PS** mining, is defined in detail in this chapter, with search operators and the metaheuristic itself. Additionally, there are additional non-trivial post-processing steps which aid the use of **PSs** as explanations.

## 6.2 SUMMARY OF METHODOLOGY

The key steps of **PS** mining are as follows:

1. A metaheuristic is executed to solve the original optimisation problem, and the solutions it evaluated are stored as a **reference population**, denoted as  $P_{Ref}$ .
2. The possible types of **PSs** are decided, by selecting their search space. For local explanations (i.e. for a specific solution), the **Subset-of-Solution** space is to be used. For global explanations, the **Any-Pattern** space is used instead.
3. The **objectives** to be optimised are decided. In particular, the user can choose what they consider as “an indication of good fitness”.
4. Having set the search space and objectives, the **PS** mining task can be solved by a **MO**: NSGA-II.
5. The final Pareto front of **PSs** produced by the **MO** optimisation algorithm is kept as the set of optimal **PSs**. Multiple **MO** runs may be executed, and the results are aggregated.

After these steps, the **PSs** are **post-processed**:

1. The quantity of items in the Pareto Front might be excessive for the user, so it can be reduced by using **clustering**
2. Each item is analysed to determine its **effect on fitness**. This is done via descriptive statistics and statistical tests
3. Considering the proxy-functions decided by the user (which are likely to relate to the fitness function), the system finds the average value for each pattern and determines if they are relatively extreme. The proxy-functions which are sufficiently extreme are called **descriptors**, and are presented to the user.

The structure of this chapter is as follows:

- Section 6.3 describes the **motivation** behind the process described above, and why a **heuristic method** is unlikely to produce satisfactory results.
- Section 6.4 discusses the **acquisition of the reference population** ( $P_{Ref}$ ), and why this approach was used.
- Section 6.5 defines the **search operators** used within the metaheuristic that will find the optimal **PSs**. For the different search spaces, different operators are implemented.

- Section 6.6 discusses the **multi-objective search** method, and the implications of its usage. This section also contains discussions on Pareto-fronts and how the system has changed over the course of the research process. A summary of how this process was tuned (in terms of parameters) is also presented.
- The **PS objectives** are described, organised in 3 main sections and with additional discussions
  - Section 6.7 contains the definitions of **PS** objectives which are **indicators of high fitness**, such as **mean-fitness**.
  - Section 6.8 defines **objectives that measure how simple a pattern is**, so that it is easily understood, such as **\*-count simplicity**
  - Section 6.9 defines **atomicity** by building on top of **linkage learning**.
  - Section 6.10 discusses and defines **other search metrics** which may be used to describe what makes a **PS** desirable.
  - Section 6.11 shows the **outcomes of picking certain combinations of objectives**, on various benchmark problems. This section is to further show that there is no “one-size-fits-all” combination of search metrics.
- Section 6.12 describes the main **post-processing** step: descriptors.
- Section 6.13 briefly discusses the optional step of **PS** clustering.
- Section 6.14 shows empirical results of the **runtime** of **PS** mining.

### 6.3 MOTIVATION BEHIND THE OVERARCHING METHODOLOGY

The procedures described above, based on a metaheuristic to find **PSs**, is not the only possible way to obtain patterns. The following paragraphs will discuss why other methods are not expected to be satisfactory.

The reasons presented here were used as a guidance in the research process which eventually resulted in our final metaheuristic-based methodology. They are not formal guarantees, and instead they were used as indicators for what methods were worth investigating. The ideas presented revolve around one idea: since we are interested in optimisation problems which *can only be solved* by a metaheuristic, and because the **PS** mining task is as hard that that original problem, only another metaheuristic process can successfully find desirable **PSs**.

**WHY NOT A HEURISTIC?** Another possible approach would be to use a **heuristic** which constructs patterns, perhaps using a greedy approach where smaller patterns are gradually assembled into bigger patterns.

It is expected that this method will not be able to produce patterns that are satisfactory, and that only another search method would be able to find them. This can be shown informally via *reductio ad absurdum*:

1. Assume that the optimisation problem being investigated cannot be solved well by a heuristic, which is why it is being solved by a metaheuristic.
2. Assume (by absurd) that there is a heuristic which finds satisfactory **PS**.
3. One can construct another heuristic to combine the various **PSs** that have just been found, with the aim of finding good solutions.
4. If Step 2 produced optimal patterns, the second step should result in the optima. Overall, the combination of *heuristic* procedures was able to solve the original optimisation problem.
5. **Contradiction:** the original problem was solved heuristically, contradicting the first assumption.

This is an informal proof, and the first assumption is especially important: our methods are well justified for scenarios where a metaheuristic is already necessary.

WHY NOT A MODEL? Another possible approach would be to train an **ML model** to predict the fitness of solutions. The model is then mined to obtain the patterns, with a heuristic. Similar approaches were investigated in [278, 247], where a surrogate model is used to obtain variable importance values.

This approach roughly falls again in the category of heuristics, although one could argue that the process of building a model could be stochastic enough as to avoid the assumption on point 2 of the previous proof. An example of such a process is a Learning Classifier System, where rules which resemble **PSs** are evolved gradually to make a model.

Such an approach would work, but the steps involved would essentially re-create our methodology but indirectly and with less flexibility.

Instead, we choose to evolve the explanations directly and avoid introducing unnecessary modelling steps.

WHY USE A METAHEURISTIC TO FIND PARTIAL SOLUTIONS? Following from the discussion on heuristics and models, a metaheuristic is the method used in this research as a tool to find useful **PSs**.

This has a practical advantage: it only requires for the objectives to be defined, and there there is no assumption of what **objectives** the explanations will need to satisfy. Thus the same methodology can work for many users with different aims, with little modification.

As an example, in the Recommender Systems research (which will be discussed in Section 8.2, different metrics were used in order to produce explanations more aligned with stakeholder intentions. These were not envisioned at the start of our research, but were integrated without even needing to know what the objectives were measuring.

Moreover, a multi-objective metaheuristic is able to approximate the Pareto front, rather than enforcing an arbitrary trade-off. Trade-off between understandability and accuracy is an inevitable reality of our explanations, and a search method can find a comprehensive set of choices which can later be selected from.

Our methodology could have been implemented as a problem-specific and metaheuristic-specific algorithm (for example, it is known that **BT** is solving the Staff Rostering (**SR**) problem using **SA**). This was not pursued as it was seen more fruitful and conceptually simpler to focus on problem-agnostic and metaheuristic-agnostic approaches.

This is why our procedure makes use of something which is easily generated in every metaheuristic: a collection of evaluated solutions (the **reference population**, stored in a file). This “static” approach forces our explanations to be post-hoc, but also makes it much more widely applicable and proved to be essential in broadening the scope to include Recommender Systems (Section 8.2).

WHY NOT EVOLVE A SET PATTERNS? The aim of this research is to generate a *set* of patterns, rather than just one, to account for cases where there is more than one trait contributing to desirable solutions. In our case, for each **PS** we obtain an explanation, but we could have attempted to create a system where there is one multi-**PS** explanation which describes the problem (or the optima) as a whole.

While this is likely achievable, it was not considered because the level of complexity required to evolve these **PS**-sets would have been too high for the resulting reward:

- **The results would be confusing:** a **MO** search over **PS**-sets would have produced a Pareto front of *sets* of solutions. The additional effort required by the user to process this information is unlikely to be acceptable.
- Evolving a set of **PSs** is complex: the operators required and the objectives used would need to be much more complex than what we propose. Moreover, the task would be more challenging for the multi-objective solver, and so the system would likely converge much more slowly.
- **The same results are achieved using our methods already:** the prediction model shown in Section 8.3 are essentially a collection of **PSs** which predict solution fitness. In this sense, they are already a concise description of the overall problem.

#### 6.4 ACQUISITION OF THE REFERENCE POPULATION

A **reference population** ( $P_{Ref}$ ) is a collection of solutions with pre-computed fitness values. Their order is not relevant, so there is no concept of generations or convergence. In fact, the source of the solutions is not really relevant: while for this thesis the source is almost always a metaheuristic, this is not a requirement because there are very few

assumptions on the distribution of solutions. If a  $P_{Ref}$  can be obtained, the **PS** mining methodology can be applied. This “**static dataset**” has many advantages:

- **Allows for PS-based explanations to be applicable to many metaheuristics**, such as Simulated Annealing (which has no population) and Random Search (which does not converge). In fact, any optimiser that generates multiple solutions would also be eligible. This is in contrast to the work of Fyvie et al where generations are the basic unit [96].
- The reference population can be stored statically, even as a `.csv` file (as shown in Section 5.7). This allows for our system to be applicable with minimal modifications to the optimisation system at hand.
- In situations **where optimisation is a very slow process** (for example if a simulation is involved, or perhaps it relies on real humans), the static dataset only needs to be generated once and can be analysed later in multiple ways. This was particularly relevant for **PS4XRS** [43] (further discussed in Chapter 8).
- Since the distribution of solutions contained is not too relevant, these **solutions can be from different sources** at the same time. One may solve an optimisation problem using different metaheuristics, and obtain multiple reference populations that are concatenated into one. In fact, the set of solutions may even be manually constructed. This can be useful for **global** explanations, where a suitable set of samples is required.

## 6.5 SEARCH OPERATORS FOR PARTIAL SOLUTIONS

The **PS** mining task will use its own search operators, separate from the operators used to solve the original problem. The same operators cannot be used in both cases, since the `*` value is not present in the original search task.

This section will discuss the following search operators, used for **PS** mining:

- An **initialisation operator**, which creates patterns that are mostly wildcards.
- A **mutation operator**, which has a chance of turning a wildcard into a fixed variable, and vice-versa (uniform mutation)
- A **crossover operator**, based on uniform crossover
- A **niching operator**, based on sequential niching (many runs, each run avoids results from the previous runs)

These operators are implemented differently between local and global search for patterns, but their algorithmic structure is essentially the same.

**PSs** can be treated as standard vectors of discrete variables, so general-use operators can be applied in order to allow wide applicability. Specialised operators lead to issues related to the No Free Lunch Theorem [291].

In the case of **PS4XRS**, the context presented strong constraints that required the development of new operators in order to get significant results, which are discussed in more detail in [43] and Chapter 8.

### 6.5.1 Preliminaries: Operators used in the early stages of research

The preliminary research shown in [44] (attached in Chapter A) discusses different approaches for **PS** mining:

- start from patterns that are almost entirely wildcards, then gradually introduce fixed variables.
- start from patterns that are almost fully fixed (generally, these are based on the optimal solutions) and gradually introduce wildcards to make them more general.

- start from randomly generated patterns, and allow them to mutate freely (wildcards into fixed variables, and vice-versa).

The evaluation shown in [44] showed how the first approach is able to recover the correct **PS** from benchmark problems more frequently. These results are not included in the main body of the thesis, since they contain a methodology which was replaced with the approach shown in this chapter. Despite this, the results were essential for developing the operators which will be described in the rest of this chapter.

## 6.5.2 Operators for Any-Pattern (global) search of Partial Solutions

The operators described here are the basis of all the other operators used for **PSs** in this research.

### 6.5.2.1 Initialisation operator

This operator was developed to produce patterns which are mostly wildcards, but some thought was put into how to define this exactly. Every possible initialisation operator would have presented a **bias** in how the patterns are generated, for example in how many fixed variables are present on average.

Consider for example the naive approach where every variable has a 50% chance of being **\***, and a 50% of being a fixed value (and each possible value has the same probability). If there are  $L$  variables, it is expected that the patterns will have  $\frac{L}{2}$  fixed variables on average, which is higher than desirable (e.g. if  $L = 100$ , then patterns will be initialised with 50 fixed variables in general).

Instead, the operator was designed in order to better align with human understanding, by making an initialisation operator that produces patterns with **7** fixed variables on average. This is informed by **Miller’s Law**, where information a human can store  $7 \pm 2$  units of information in their short-term memory [194].

Patterns with fewer fixed variables should be more common, since the aim is simplicity. Therefore, the final algorithm is constructed to follow a **geometric** distribution in the quantity of fixed variables observed, starting from 1 (so that the “empty” pattern is not produced). Algorithm 9 shows how we implemented an operator with the aforementioned properties.

---

#### Algorithm 9 Generate Random Pattern

---

```

1: function SAMPLE_RANDOM_PS_AP
2:   probability_of_fixed_variable  $\leftarrow$  0.83
3:    $x^\psi \leftarrow (* \dots * * *)$ 
4:   keep_going  $\leftarrow$  true
5:   while keep_going do
6:     variable  $\leftarrow$  random.randrange( $L$ )
7:     value  $\leftarrow$  random.choice(options(variable))
8:      $x^\psi[\text{variable}] \leftarrow$  value
9:     keep_going  $\leftarrow$  random() < probability_of_fixed_variable
10:  end while
11:  return  $x^\psi$ 
12: end function

```

---

A geometric distribution is achieved by using a simple **while** loop in Algorithm 9. The quantity of “successful events” (a variable is set to a fixed value) in consecutive trials (where each trial has a probability  $p$  of succeeding) follows a geometric distribution, and the average quantity of successes is  $\frac{1}{1-p}$ . [256] This formula can be reversed to find the value of  $p$  that yields an average of 7 successes (accounting for the forced initial success), which is  $p = 0.83$  (hence why the magic number in this operator). This constant is independent of the total quantity of variables.

---

**Algorithm 10** Get Mutated Pattern

---

```
1: function MUTATE_PS_AP( $x^\psi$ )
2:   mutation_probability  $\leftarrow 1/L$ 
3:    $y^\psi \leftarrow x^\psi.\text{copy}()$ 
4:   for index, value  $\in \text{enumerate}(y^\psi)$  do
5:     if random()  $<$  mutation_probability then
6:       if value == * then
7:          $y^\psi[\text{index}] \leftarrow \text{random.choice}(D_{\text{index}})$ 
8:       else
9:          $y^\psi[\text{index}] \leftarrow *$ 
10:      end if
11:    end if
12:  end for
13:  return  $y^\psi$ 
14: end function
```

---

#### 6.5.2.2 Mutation operator

The mutation operator shown in Algorithm 10 is essentially **uniform mutation**, where the mutation probability is  $\frac{1}{L}$  as is commonly recommended [208]. This could be called a bit-flip operator, but instead of flipping between **true** and **false** it flips between \* and non-\* values (which themselves are picked at random uniformly).

This kind of mutation is chosen for its generality, as it makes the least assumptions about interaction effects between variables [269].

This operator also has a bias: if there was no selective pressure and the patterns were to mutate freely for generations, they will have  $\frac{L}{2}$  fixed variables on average over time. Although this is not ideal, it is an effect that would only be observed in an unrealistic scenario. Moreover, this is again a case where bias is inevitable.

#### 6.5.2.3 Crossover operator

Our crossover operator is based on **uniform crossover** (again to avoid biases), and is defined in Algorithm 11.

---

**Algorithm 11** Get Crossed Patterns

---

```
1: function CROSSOVER_PS_AP( $a^\psi, b^\psi$ )
2:    $c^\psi, d^\psi \leftarrow a^\psi.\text{copy}(), b^\psi.\text{copy}()$ 
3:   for index  $\in \text{range}(L)$  do
4:     if random()  $<$  crossover_probability then
5:        $c^\psi[\text{index}], d^\psi[\text{index}] \leftarrow d^\psi[\text{index}], c^\psi[\text{index}]$ 
6:     end if
7:   end for
8:   return  $c^\psi, d^\psi$ 
9: end function
```

---

This operator could be improved for domain specific applications by introducing linkage information (which was the case for Partial Solutions For Recommender Systems (PS4XRS)), but our implementation in Algorithm 11 aims to avoid assumptions of relationships between variables.

#### 6.5.2.4 Niching / Constraint

Maintaining diversity is often required in multi-objective search in order to obtain a comprehensive set of results. **Niching** is a way to achieve this, where individuals which are too similar are removed from the metaheuristic's population.

This approach was adopted for PS mining, to account for situations where multiple different patterns are sought, and was first implemented in [46] as “sequential decision space niching”. The meaning of these terms is:

- **sequential**: there are multiple metaheuristic runs, and each run is given the patterns found in previous runs. The niching operator will negatively affect the survival rate of patterns that are too similar to the previously found ones.
- **decision space**: in traditional multi-objective search methods, niching is typically implemented over objective space: the result should have a wide range of *objective values*. In our case, it is more important to have patterns that are different in their variables (this being decision space). This is the niching approached used by Omni-Optimizer [78].

Niching, specifically in NSGA-II, is a mechanism that is used as a tie-breaker during selection. We opted for an approach which is essentially a constraint: solutions which break the constraint - needing to be different enough from previous runs - are forcefully removed from the population.

The operator we developed, shown in Algorithm 12, can be summarised as:

- Run multiple PS mining runs, each with a different “niching constraint” based on the results of the previous runs. In the first run, the constraint is turned off.
- A distance metric is defined over PSs, as shown in Algorithm 12. Patterns are “too similar” when their distance is below a threshold. This threshold is chosen to be 0.5, so that PSs are too similar if more than half of their fixed variables are the same.
- Define a constraint operator for PS runs. A pattern is allowed if its distance from all of the previously found patterns (Pareto fronts of the previous PS mining runs) is above the aforementioned threshold.

### 6.5.3 Operators on Subset-of-Solution (local explanations) for Partial Solutions

The operators implemented for local search are essentially the same as in Section 6.5.2, and their implementation is summarised as follows:

- Sampler (Algorithm 13): a generated pattern’s fixed values are taken from a given solution rather than being randomly generated.
- Mutation (Algorithm 14): when a variable is turned from a wildcard into a fixed variable, the fixed variable is chosen from the original solution.
- Crossover and Niching: no difference

Their exact implementation is reported here for completeness.

---

**Algorithm 12** Distance and Allowed Pattern Check

---

```
1: function DISTANCE( $x^\psi$ , old_pattern)
2:                                     ▷ Returns the ratio of shared fixed variables
3:   shared_fixed  $\leftarrow$  0
4:   qty_fixed_in_old  $\leftarrow$  0
5:   for  $i \in \text{range}(L)$  do
6:     if  $x^\psi[i] \neq -1$  and old_pattern[i] ==  $x^\psi[i]$  then
7:       shared_fixed  $\leftarrow$  shared_fixed + 1
8:     end if
9:     if old_pattern[i]  $\neq -1$  then
10:      qty_fixed_in_old  $\leftarrow$  qty_fixed_in_old + 1
11:    end if
12:  end for
13:  if qty_fixed_in_old == 0 then
14:    return 0
15:  else
16:    return shared_fixed/qty_fixed_in_old
17:  end if
18: end function

19: function PATTERN_IS_ALLOWED( $x^\psi$ , old_patterns, min_distance = 0.5)
20:   for old_pattern  $\in$  old_patterns do
21:     if DISTANCE( $x^\psi$ , old_pattern) < min_distance then
22:       return False
23:     end if
24:   end for
25:   return True
26: end function
```

---

---

**Algorithm 13** Generate Random Pattern on Subset-of-Solution

---

```
1: function SAMPLE_RANDOM_PS_SOS
2:    $s \leftarrow$  the solution to be explained
3:   probability_of_fixed_variable  $\leftarrow$  0.86
4:    $x^\psi \leftarrow (* \dots * * *)$ 
5:   continue  $\leftarrow$  true
6:   while continue do
7:     variable  $\leftarrow$  random.randrange( $L$ )
8:     value  $\leftarrow$   $s[\text{variable}]$ 
9:      $x^\psi[\text{variable}] \leftarrow$  value
10:    continue  $\leftarrow$  random() < probability_of_fixed_variable
11:  end while
12:  return  $x^\psi$ 
13: end function
```

---

---

**Algorithm 14** Get Mutated Pattern on Subset-of-Solution

---

```
1: function MUTATE_PS_SOS( $x^\psi$ )
2:    $s \leftarrow$  the solution to be explained
3:    $\text{mutation\_probability} \leftarrow 1/\text{len}(x^\psi)$ 
4:    $y^\psi \leftarrow x^\psi.\text{copy}()$ 
5:   for  $\text{index}, \text{value} \in \text{enumerate}(y^\psi)$  do
6:     if  $\text{random}() < \text{mutation\_probability}$  then
7:       if  $\text{value} == *$  then
8:          $y^\psi[\text{index}] \leftarrow s[\text{index}]$ 
9:       else
10:         $y^\psi[\text{index}] \leftarrow *$ 
11:      end if
12:    end if
13:  end for
14:  return  $y^\psi$ 
15: end function
```

---

As defined in Chapter 5, the **PSs** of interest are the ones which satisfy multiple objectives, such as having few fixed variables while being associated with high fitness solutions. The objectives themselves are not fixed, and rather they may change between users.

Therefore, the **PS** mining task is a multi-objective problem, which will be solved using a **MO** search method. Determining which one is the most appropriate required a problem with ground-truth patterns, and determining which metaheuristic finds them most often when given a fixed evaluation budget.

### 6.6.1 Experiments to determine the most suitable Multi-objective method

This evaluation was carried out in [46], where synthetic instances of the Staff Rostering problem were used so that ground truth **PSs** could be used to score the outcome of a **PS** search run. These **benchmark problems** are constructed to be mathematically equivalent to RoyalRoad (certain employees must have the same rota) and Graph Colouring problems (certain employees must have different rotas).

Here, equivalence means that there is a one-to-one monotonic mapping between the fitness values of the original problems and their **SR** counterpart. The problems are called SR-RR and SR-GC respectively. A complete definition is found in Chapter E. The “target” or “ground truth” **PSs** for these problems is the same as what would be seen in their original counterpart:

- SR-RR is composed of 5 groups of workers, each having 4 employees. Since RoyalRoad is a binary problem, each employee has 2 possible rotas. A group of workers contributes positively to fitness only if they are *all* assigned to their second rota. The target **PSs** are 5, each being a group of workers being all being set to their second rota.
- SR-GC has 5 independent groups of workers, each having 3 employees. Like a Graph Colouring problem with 3 colours available, each employee has 3 possible rotas. A group of workers contributes positively to fitness when all its employees are assigned to *different* rotas from each other. There are 30 target **PSs**, 6 for each group (because there are 5 possible ways in which the workers can have different rotas from each other).

The use of **SR** equivalents of existing benchmark problems is due to how at this stage in the research process, the **SR** problem was our main concern. Additionally, this was to show that there are **SR** problem instances which do display interaction effects. Thanks to the mathematical equivalence of the problems, the results are expected to be identical on the original benchmark problems.

The evaluation considered 4 possible multi-objective methods: MOEA/D [297], NSGA-II [72], NSGA-III [76] and SMS-EMOA [24]. To structure the objective space, both MOEA/D and NSGA-III utilized reference directions generated via Das-Dennis method with 12 partitions [69]. Additionally, the neighbourhood size in MOEA/D was dynamically assigned to equal the total number of decision parameters in the search space. Across all four algorithms, the operators used were as defined in Section 6.5, with crossover probability being set to 0.5.

Moreover, the evaluation also aimed at determining whether it is beneficial to have few runs with a large evaluation budget, or to carry out many smaller ones, which relates to the sequential niching operator. A **PS** mining run is considered successful if all the “ground truth” **PSs** are found in the union of the Pareto fronts of the runs. These ground truth patterns are built into the problems, as specified in Chapter E. In our evaluation, runs were repeated 100 times, and we counted how many of those were successful.

All runs had the same overall evaluation budget of 10000, but the population size and budget-per-run were varied between (100, 1000) (10 small runs), (200, 5000) (2 runs), (500, 10000) (a single large run).

A run was considered **successful** if *all the target PSs* were present in the Pareto front produced by the **PS** process. Even one missing target **PS** is considered a failed run.

| Problem | Pop.<br>Size | Evaluation<br>Budget | MOEAD | NSGA-II    | NSGA-<br>III | SMS-<br>EMOA |
|---------|--------------|----------------------|-------|------------|--------------|--------------|
| SR-RR   | 50           | 1000                 | 42%   | <b>93%</b> | 25%          | 87%          |
|         | 100          | 5000                 | 26%   | <b>86%</b> | 41%          | 61%          |
|         | 200          | 10000                | 21%   | <b>77%</b> | 29%          | 48%          |
| SR-GC   | 50           | 1000                 | 23%   | <b>68%</b> | 32%          | 67%          |
|         | 100          | 5000                 | 10%   | <b>27%</b> | 26%          | 20%          |
|         | 200          | 10000                | 7%    | <b>18%</b> | 14%          | 12%          |

Table 6.1: Table showing the results of the comparison between different Multi-Objective evaluation methods. The percentage shows how often the procedure was able to find all of the expected **PSs**. Figures in bold show the best performing **MO** method for every row.

The results (shown in Table 6.1) indicate that **NSGA-II** (of the Non-Dominated Sorting Genetic Algorithm family) performs better than the other methods on benchmark problems, with the best results being observed with many smaller runs. This result is not entirely surprising, since most multi-objective search methods are made for continuous problems and contain specialisations [73]. **NSGA-II** makes the least assumptions and thus it works best on our combinatorial **PS** search task.

**NSGA-II** exhibits a tendency to be biased towards “simpler” results when applied in Genetic programming [179], and similar results are expected in our implementation. Although this does introduce bias, it is not in a direction which clashes against our intentions, and thus we opted to use a traditional formulation of **NSGA-II**.

**NOT PYMOO** Throughout most of the research, the implementation of **NSGA-II** was taken from **Pymoo** [26], but this was later found to be bloated and inconsistent with the original definition of **NSGA-II**. We implemented a version which is more aligned with the methodology discussed in the original **NSGA-II** paper [72]. This change is not expected to have significant effects on the performance of the algorithms themselves, but rather it reduced attrition when working on the implementation of the **PS** mining task. **PYMOO**’s implementation involves many additional steps which cancel each other out and were removed without consequences.

### 6.6.2 Budget and Other Parameters

The search process requires specifying parameters such as the evaluation budget and population size.

The evaluation shown in Table 6.1 does provide some insight, but it is likely that there is no universally sufficient set of parameters due to how the complexity of the **PS** search task is determined by the complexity of the original optimisation problem.

A simple rule was used throughout this research: **use the same hyper-parameters that were used for solving the original task.**

### 6.6.3 Resulting Pareto Front

The result of the multi-objective search process is an approximation of the *Pareto front*: a set of non-dominated **PSs**. Although having a range of patterns might seem inconvenient, it provides insight into the trade-off between *understandability* and *effect on fitness*.

**SELECTING A SINGLE SOLUTION** In many cases, it is necessary to select a single **PS**, or at least to provide an ordering for them (e.g., when making a model and only one **PS** can be picked). For such instances, the approach suggested is to order the results “from least to most extreme”.

Informally, each candidate has a ranking for each objective (e.g., [**\*** \* 1 1] might be 1st in mean fitness, 3rd in simplicity, and 7th in atomicity). The “worst rank” of a pattern is the “worst” of its rankings, and in the previous example it would be “7th”. A pattern which is generally good in all metrics will have a lower “worst”, whereas pattern which ranks poorly in at least one metric has a greater “worst rank”. This mechanism is widespread in decision-making theory, under the name “**minimax** regret” [236].

A procedure which sorts the **PSs** using this criterion is shown in Algorithm 15. When one pattern is requested, we sort the given set of patterns from best to worst and return the single best.

---

**Algorithm 15** Sort Population by Max-Min Tradeoff

---

```
1: function GET_WORST_RANK(sol_objs, pop_objs)
2:   num_objs  $\leftarrow$  length(sol_objs)
3:   ranks  $\leftarrow$  zeros(num_objs) ▷ Initialize rank array
4:   for  $i \leftarrow 1$  to num_objs do
5:     for each other_objs in pop_objs do
6:       if sol_objs[ $i$ ] < other_objs[ $i$ ] then
7:         ranks[ $i$ ]  $\leftarrow$  ranks[ $i$ ] + 1
8:       end if
9:     end for
10:  end for
11:  return max(ranks)
12: end function

13: function SORT_BY_WORST_RANK(pop, pop_objs)
14:   worst_ranks  $\leftarrow$  []
15:   for each row in pop_objs do
16:     wr  $\leftarrow$  GET_WORST_RANK(row, pop_objs)
17:     Append wr to worst_ranks
18:   end for
19:   ▷ Sort individuals ascending by their worst rank
20:   sorted_indices  $\leftarrow$  argsort(worst_ranks, ascending=True)
21:   sorted_pop  $\leftarrow$  [pop[ $i$ ] for  $i$  in sorted_indices]
22:   return sorted_pop
23: end function

24: function GET_BEST_TRADEOFF(pop, pop_objs)
25:   sorted_pop  $\leftarrow$  SORT_BY_WORST_RANK(pop, pop_objs)
26:   return sorted_pop[0]
27: end function
```

---

This section will be the first to discuss the objectives that can be used to search for **PS**. This section focuses on quantifying the effect that a pattern has on the fitness of the solutions that contain it. The effect on fitness could be measured in two ways:

1. A “**directional measurement**”, where we measure whether the fitness of solutions containing the pattern is **better** than those that do not.
2. A “**non-directional measurement**”, where we measure how much the fitness is affected by a pattern, which could be both a positive or negative effect.

This is similar to statistical testing, in which a test that compares the mean of two averages can measure if one is greater than another (single-tailed test) or if the two averages are different (two-tailed test). For both situations, there are two approaches to detecting this effect:

1. an “**independent samples**” measurement, where we partition the  $P_{\text{Ref}}$ ’s solutions based on whether the pattern is present or not. Then, we compare the fitness values of these groups.
2. a “**paired samples**” measurement, where we compare the fitness of a solution *before* and *after* the pattern was applied onto it.

Note that the terminology was borrowed from statistics for convenience, and the quality metrics don’t necessarily need to be based on statistical tests.

Finally, there are two kinds of scalar values which quantify an effect on fitness:

- **effect size**: how strongly the pattern influences fitness, the magnitude of the change.
- **a probability**: the likelihood of the pattern having an effect. Usually, this is a p-value, which also quantifies if there is *sufficient evidence* of the effect on fitness.

Therefore, a pattern might have a large *estimated* effect size, but low probability of being provably beneficial. A user should be aware of this difference.

**CONSIDERATIONS** When picking the quality metric, there are some practical **considerations**, which are listed below.

1. The  $P_{\text{Ref}}$  is rarely an enumeration of the search space so it will generally **not contain all possible solutions which match a pattern**. This can be a problem for paired samples tests, and forces either to evaluate new solutions, or to use only paired samples already present in the data. Both approaches are rather computationally slow (since the entire  $P_{\text{Ref}}$  needs to be processed for every **PS**), and might not be usable for the purposes of an interactive tool, therefore a paired sample fitness quality test would be too slow in practice.
2. Because the  $P_{\text{Ref}}$  is rarely uniformly distributed, and due to its origin (from a metaheuristic) it is **often biased**. A lot of solutions will be similar to the optimum (due to convergence), or there might be popular genes due to *hitch-hiking* (discussed in Section 2.3.11.2).
3. When using a non-directional measurement, a pattern which negatively influences fitness can also score high. In these cases, the **lack of the pattern** is beneficial, and might thus be insightful to the user, but in other cases it is not very intuitive.
4. The **bias in the  $P_{\text{Ref}}$  might be used as a proxy for fitness quality** (or as a way to investigate metaheuristic behaviour). In these cases, quality of a pattern is not based on its fitness, but in how common it is within the  $P_{\text{Ref}}$ .

Due to these considerations, many metrics to measure “effect on fitness” were developed and are reported in this chapter.

As a shorthand in these formulas, we often use the notation  $P_{\text{match}}$  and  $P_{\text{notmatch}}$  to mean “solutions which match the pattern” and “solutions which do **not** match the pattern”. Whenever these are used, the reader can assume that the objective is influenced by the  $P_{\text{Ref}}$ .

**MEAN-FITNESS** This metric serves as a straight-forward and computationally lightweight approach to measuring fitness quality. More sophisticated approaches are presented in this section, but **mean-fitness** is the most generally applicable.

**Definition:** the **mean-fitness** of a **PS** is the average fitness of the solutions which it matches.

$$\text{mean\_fitness}(x^\psi) = \begin{cases} \text{avg}(F(P_{\text{match}})) & \text{when } |P_{\text{match}}| > 0 \\ -\infty & \text{otherwise} \end{cases} \quad (6.1)$$

Using our previous classifications, it is a directional, independent samples, effect size measurement. This metric is to be maximised (assuming that the original fitness is to be maximised). Note how, if there are zero matching solutions, then a low value is produced - this is rarely encountered throughout a real search for patterns because of other metrics that promote simpler patterns which will have more samples.

**Pros and Cons:** This metric can be easily skewed when there are **few solution samples** for a pattern. Although a threshold quantity of samples could be manually set (so that  $-\infty$  is returned when  $|P_{\text{match}}| < \text{threshold}$ ), a more robust approach is to use a statistical test, as described next, but these also have their issues.

In the rest of the thesis **mean-fitness** is the most commonly used metric to measure fitness quality of a pattern for this research because of how relatively simple it is to understand at a glance.

**MANN-WHITNEY-U STATISTICAL TEST** A user might require for the **PSs** produced to be provably beneficial to fitness, which can be assessed by using statistical tests. Rather than doing this after **PS** mining, it is possible to use the p-values from the tests as an objective to quantify the effect on fitness with the group of solutions in the  $P_{\text{Ref}}$  containing a given pattern.

**Definition:** The fitness values of  $P_{\text{match}}$  and  $P_{\text{notmatch}}$  can be used for an **independent samples statistical** test, with the p-value being an objective to be minimised.

A Mann-Whitney-U test is used, due to it being non-parametric (no assumptions are made on  $P_{\text{Ref}}$ ), and it can be both directional and non-directional. The test determines if there is a difference in average fitness between solutions *with* the pattern, and the solutions *without* the pattern.

At first, this would seem like an improved version of **mean-fitness**, since a pattern will have a low score by when not enough samples exist to prove its effect. Differently from **mean-fitness**, it does not reflect how *strongly* the pattern affects fitness, just how likely it is to be having an effect.

When defining this metric, the following issues need to be considered:

- **Automated p-hacking:** the metaheuristic is able to find patterns that “accidentally” produce a low p-value, without actually having a high effect size.
- **Minimising to zero:** a p-value of 0.003 is more than sufficient, but the search will easily find patterns with a p-value smaller than 0.0001. Since one is lower than the other, the algorithm might discard patterns with satisfactory test results.

To resolve these issues, a threshold is put on the metric: any p-value below 0.05 is set as zero. This means that any patterns that are “provably beneficial” (below a significance of 0.05) will score equally.

Using these considerations, we get the following metrics, **single\_indep\_test** for a directional test, and **two\_indep\_test** for a non-directional test:

$$\begin{aligned} \text{MWU\_p\_value}(x^\psi, \text{isSingleTailed}) = \\ \text{MannWhitneyUTest}(F(P_{\text{match}}), F(P_{\text{notmatch}}), \text{isSingleTailed}) \end{aligned} \quad (6.2)$$

$$\text{threshold}(\text{orig}, t) = \begin{cases} \text{orig} & \text{when orig} > t \\ t & \text{otherwise} \end{cases} \quad (6.3)$$

$$\text{single\_indep\_test}(x^\psi) = \text{threshold}(\text{MWU\_p\_value}(x^\psi, \text{true}), 0.05) \quad (6.4)$$

$$(6.5)$$

**Pros and Cons:** For a user who requires the explanations to be provable, using this metric aligns with their needs.

Unfortunately, carrying out a statistical test is computationally slower than **mean-fitness**, and causes the system to be less usable as an interactive tool. Additionally, the problem of automated p-hacking is not entirely resolved: a **PS** might be *just below* the significance threshold because it relates to solutions which *by chance* have high fitness.

**PAIRED SAMPLES TESTS** If independent samples test provide a metric of how strongly patterns correlate with high fitness, paired samples approaches measure the **causal** relationship between patterns and fitness. Therefore, they are even more robust than independent sample tests, but they are not feasibly applicable due to how **slow** they would be. This section discusses their hypothetical implementation.

**Definition:** Assume that, for any given **PS**  $x^\psi$  there is a way of obtaining a set pairs of solutions “before and after” the pattern was applied - let us call it **paired\_samples** $x^\psi =$  a collection of pairs  $(a, b)$  such that  $\text{match}(a, x^\psi) == \text{False}$  and  $b == \text{apply}(a, x^\psi)$ . (The implementation of this function is what prevents using this metric).

Using the fitness values of these paired solutions, a Wilcoxon Signed Rank test could be used (because it is a non-parametric paired samples comparison of average). [226].

Similarly to the independent-samples metrics discussed above, the p-value would be minimised.

**Pros and Cons:** This approach cannot be used in reality as a search objective. The reason lies in the need for the paired samples: they are very computationally expensive to obtain. There are two ways of obtaining these pairs, and both are highly inefficient:

- **Evaluate new solutions:** Randomly pick a set of solutions, and for each obtain a before-after pair by applying  $x^\psi$ . This quantity of solutions should be big enough to give power to the statistical test. These new solutions need to be evaluated using the fitness function, and therefore each evaluation of  $x^\psi$  involves many evaluations of the fitness function, making it unfeasible.
- Find pairs within the  $P_{\text{Ref}}$  which happen to be pairs of before-after solutions. This is slow, and it is highly unlikely to yield enough pairs for the statistical test to have sufficient power.

Rather than being used as an objective, this statistical test could find use as a post-processing tool in future work.

**WEIGHTED VARIANCE** This metric was developed for the research which uses **PSs** within **decision trees**, and comes directly from CART [33]. The generation of a decision tree is traditionally greedy and recursive: the dataset is partitioned, and each partition is picked for being optimal with respect to a metric.

In the case of a decision tree for a regression task, the metric to be maximised is the **reduction in variance**: by partitioning a set of data-points, the sum of their variances should be much lower than the variance of the original collection.

In our case, a pattern is used to partition the  $P_{\text{Ref}}$  into  $P_{\text{match}}$  and  $P_{\text{notmatch}}$ , each having a variance in their fitness values. Let **var** be the function which calculates the **variance** of a collection of scalar values. One could calculate the variance of the partitions as  $\text{var}(F(P_{\text{match}})) + \text{var}(F(P_{\text{notmatch}}))$ , but this will be biased by skewed partitions (perhaps  $P_{\text{match}}$  has low variance because it has one solution!). Therefore, **weighted variance** is calculated, where the weights are based on the sizes of the splits:

$$\begin{aligned} \text{weighted\_variance}(x^\psi) &= \frac{|P_{\text{match}}|}{|P_{\text{Ref}}|} \text{var}(F(P_{\text{match}})) \\ &+ \frac{|P_{\text{notmatch}}|}{|P_{\text{Ref}}|} \text{var}(F(P_{\text{notmatch}})) \end{aligned} \quad (6.6)$$

From this, one can calculate the **variance\_reduction**:

$$\text{variance\_reduction}(x^\psi) = \text{var}(F(P_{\text{Ref}})) - \text{weighted\_variance}(x^\psi) \quad (6.7)$$

Compared to **weighted\_variance**, all this last formula does is add the constant value  $\text{var}(F(P_{\text{Ref}}))$  and changes the sign of Equation (6.7). Therefore, it is sufficient to minimise **weighted\_variance** in order to maximise **variance\_reduction**.



PSs aim to be simpler to understand than solutions because they contain less information: a pattern only shows the variables which are important, and the rest is irrelevant for the pattern being looked at.

For such patterns to be found, it is necessary to define an objective to capture this, and this section will propose two main approaches. The first is just “having few fixed variables”, while the second one defines it in terms of linkage.

**\*-COUNT SIMPLICITY** This metric was used throughout this research, and it is the simplest to define. It is the quantity of (\*) present in a pattern, which is to be **maximised**:

$$\text{*\_count\_simplicity}(x^\psi) = |\{i \in [1 \dots L] \text{ s.t. } x_i^\psi = *\}| \quad (6.8)$$

(The empty pattern (\*\*...\*\*) is optimal for this metric.)

This objective has the **advantage** of being conceptually simple, especially since the “simplicity” of a pattern can be estimated at a glance even by humans. It is not dependent on the  $P_{\text{Ref}}$  and it is constant across different optimisation problems.

There are also **disadvantages**:

1. This concept of simplicity does not necessarily encapsulate what it means for a pattern to be “easy to understand” to a human. A small but incomplete pattern might actually be harder to understand - the “simplest” pattern is also the one that carries the least meaning.
2. The metric does not help the user determine what is the minimum level of complexity needed to understand the optimisation problem. For a RoyalRoad problem with cliques of size 4, the smallest useful pattern is (1111\*\*\*\*), but there is always a pattern which is simpler and still associated with high fitness, such as (11\*\*\*\*\*).
3. Including this metric as an objective in the PS mining task significantly increases the size of the Pareto front (which might overwhelm the user, discussed in Section 6.11). Given any pattern that is optimal in the other objectives, any of its “sub-patterns” will be in some way optimal or nearly optimal. For example, (1111\*\*\*\*) (with high mean fitness and high atomicity, for example) is mutually non-dominated by (\*111\*\*\*\*), (\*\*11\*\*\*\*), ...

The first issue is complex - understandability is a user-specific concept and thus there cannot be a universal objective in this sense. **\*-count simplicity** serves as a “close-enough” proxy. The second and third issue lead to a discussion on what makes a pattern “whole enough to be useful” while being “as small as it can be”. This is resolved in this work by using **linkage information**: a pattern is “complete enough” when it has high **independence**(defined below), and it is “as small as possible” if it has high **atomicity**.

**INDEPENDENCE** This metric is born out of the issues discussed for **\*-count simplicity**. Rather than defining a pattern as “simple if it is small”, this objective defines simplicity as “fewest uncontrolled dependencies”.

**Linkage learning** is defined in Chapter 7, but a surface level understanding is required in order to understand this section. In the relationship between solutions and their fitness, every variable has an “effect” on the fitness value. This effect of this value may also depend on the value of other variables - we say that they are linked. A linkage learning method is a way to quantify this relationship, and it implements a function called **linkage**.

Here, linkage information is used to determine if there is any “outgoing” linkage in a pattern between a fixed variable and a non-fixed variable.

Consider a RoyalRoad problem and the pattern (1 \* 1 1 \* \* \*). Between the first and fourth variable there is strong linkage (since they are within the same clique), but

there is also strong linkage between the first and the *second* (unfixed) variable. When this pattern is present in (or applied on) a solution, its effect on fitness is dependent on the value of the second variable, which is not controlled by the pattern. Compare this against (1 1 1 1 \* \* \* \*), which will always result in improved fitness in a solution which did not contain it before. We say that the first pattern is *less independent* than the second one.

One can measure how independent a pattern is by measuring the linkage of its *fixed* variables against that of its *non-fixed* variables. These linkage strengths are averaged, which gives a measure of its outward *dependence*. Because the other objectives are also maximised, we flip the sign of *dependence* and we obtain *independence* (Equation (6.9)).

Note that such an average is not defined for patterns that are fully fixed or fully unfixed, and thus the real definition will return a low number in those cases.

$$\text{independence}(x^\psi) = \begin{cases} -\infty & \text{if } \text{fix\_count}(x^\psi) \in \{0, L\} \\ \text{avg}(\{\text{linkage}(i, j) & \text{otherwise} \\ \text{s.t. } x_i^\psi \neq *, x_j^\psi = *\} \end{cases} \quad (6.9)$$

This metric is to be maximised, so that the patterns are as independent as possible. This objective is recommended over *\*-count simplicity* in some scenarios, because it more accurately evaluates what is a “useful” pattern.

A **disadvantage** is that the objective is entirely dependent on linkage information, which may be slow to calculate or even incorrect. An even more serious fault with this objective is that it will give high scores to “composite but complete” patterns, where different complete and atomic patterns are in conjunction.

For example, the pattern (1111 \*\*\*\* 1111 \*\*\*\*) contains two cliques which are fully fixed, and therefore it is technically independent. Atomicity is therefore strongly advised to be used as an “antagonist” to this metric.

**PROBLEM SPECIFIC SIMPLICITY METRICS** It is possible to define problem-specific simplicity objectives.

In the case of a Graph Colouring problem, a **PS**’s complexity may take into consideration how many colours have been used in the pattern, and in a problem where there is a spatial meaning to the variables we may consider how big their bounding box is.

Although appealing, these problem-specific definitions of simplicity are likely to introduce a **bias** into what explanations are produced: the researcher is creating a tool to create the explanation they expect already.

On the other hand, this bias might be intended - the researcher might intend to use our system as a tool to fill gaps in explanations they have already.

The concept of atomicity is one of the main contributions of this thesis, since there is relatively little literature on using linkage information for explainability.

The general aim of this objective is to quantify how much a pattern is “atomic”, which is the opposite of “composite”. Being “atomic” is a desirable property for a **PS**, because it indicates that it cannot be simplified without breaking its linkage structure. **Atomicity** encodes the human preference for “minimal” patterns. Without it, **PS** mining has a tendency to produce highly composite patterns, as these satisfy the other objectives.

**Atomicity is built upon linkage information**, which can be obtained in many ways. Atomicity is only as good as the linkage methods that it is built upon. Inaccurate linkage information leads to inaccurate atomicity, which leads to undesirable **PSs** and eventually low-quality explanations.

A full dive into the topic of linkage is deferred to Chapter 7, so that this chapter can focus on atomicity itself. This section will show one way in which linkage data is used to implement an atomicity objective, in the form of **average bivariate atomicity**.

### 6.9.1 Average bivariate atomicity

Here we define the approach developed for in this work. This method addresses the main challenge of linkage learning: the computational cost of analysing multiple variables at the same time. This is in the interest computational speed - the overall system should be fast enough to be used interactively.

**BIVARIATE PRECOMPUTED LINKAGE** Informally, the computational complexity of calculating linkage information is exponential with respect to the quantity of variables being examined at once. Measuring how linked 4 variables are is much slower than measuring the linkage of 2 variables, since there are many more value combinations to consider.

A much faster approach is to calculate all **bivariate linkage** (between every pair of variables) and then approximate higher-order linkage by using those values.

Using this technique we define atomicity as shown in Equation (6.10), where  $\text{linkage}(a, b)$  is simply accessing values in a 2d matrix.

$$\text{atomicity}_{\text{bivariate}}(x^\psi) = \begin{cases} \text{avg}(\{\text{linkage}[a, b] \mid x_a^\psi \neq *, x_b^\psi \neq *, a \neq b\}) & \text{if } \text{fixed\_count}(x^\psi) > 1 \\ -\infty & \text{otherwise} \end{cases} \quad (6.10)$$

In simpler terms, it calculates the average pairwise linkage of the fixed variables in a pattern. If there are not enough fixed variables, there is no atomicity and thus a very low value is produced. The average-pairwise-linkage is also used by many works by Thierens *et al*, such as [265, 266, 31, 263], because of its stability. Independence is also an average of pairwise linkages for this reason.

For a pattern which contains only variables that are all strongly linked to each other (a mathematical clique), then average bivariate atomicity will reflect this. When a “disconnected” variable is present, there will be more weak pairwise linkages and thus the overall average is reduced.

This approach is suitable to measure when a pattern is a clique, or the *subset of a clique*: (1\*11 \*\*\*\*) will likely have similar atomicity to (1111 \*\*\*\*).

This is the main disadvantage of atomicity: it measures “lack of excess” but disregards “incompleteness”. The metric that follows (**atomicity+independence**) is an answer to these issues.

### 6.9.2 Motivation behind atomicity

When the only search metrics being used are **\*-count simplicity** and **mean-fitness**, a Pareto Front is obtained, which spans from “simple but insignificant” to “overly complex but highly significant” patterns. The issue is that this range of patterns does not necessarily match what is meaningful to a human, and instead produces patterns that make sense mathematically but are not insightful.

Consider for example a RoyalRoad problem instance, with 2 cliques of size 4. Using **simplicity** and **mean-fitness** as metrics, the following Pareto Front is obtained (with some items removed for clarity):

|                   |                |                       |
|-------------------|----------------|-----------------------|
| (1 1 1 1 1 1 1 1) | simplicity = 0 | mean-fitness = 2.     |
| (1 * 1 1 1 1 1 1) | simplicity = 1 | mean-fitness = 1.5    |
| (1 * 1 * 1 1 1 1) | simplicity = 2 | mean-fitness = 1.25   |
| ...               | ...            | ...                   |
| (1 1 1 1 1 1 * *) | simplicity = 2 | mean-fitness = 1.25   |
| (* * * 1 1 1 1 1) | simplicity = 3 | mean-fitness = 1.125  |
| (* * 1 * 1 1 1 1) | simplicity = 3 | mean-fitness = 1.125  |
| (* * * * 1 1 1 1) | simplicity = 4 | mean-fitness = 1.0625 |
| (* * * * 1 1 1 *) | simplicity = 5 | mean-fitness = 0.5625 |
| (* * * * * 1 1)   | simplicity = 6 | mean-fitness = 0.3125 |
| (* * * * * 1 * *) | simplicity = 7 | mean-fitness = 0.1875 |
| (* * * * * * 1)   | simplicity = 7 | mean-fitness = 0.1875 |
| (* * * * * * *)   | simplicity = 8 | mean-fitness = 0.125  |

(6.11)

There are two main issues observed in the Pareto front, knowing that the patterns that matter are (1 1 1 1 \* \* \* \*) and (\* \* \* \* 1 1 1 1),

- every slight modification and combination of the target patterns is present, because they represent a non-dominated trade-off.
- of the patterns which we know are a correct explanation, every “subset” of them (where a fixed variable is replaced with \*) because they are simpler and thus they improve on one objective.

Compounded, these issues prevent the target patterns from being visible to the user. The missing component is **linkage information**: the target patterns should be immediately noticeable because they are composed of variables that *depend* on each other.

The issues are resolved through **atomicity**, where patterns which are composite (bigger than they need to be) are scored low. This is the complement to **independence** defined previously, although **atomicity** was first introduced in [44] and **independence** is introduced in this thesis.

A comprehensive review and implementation of various linkage learning methods is provided in Chapter 7. In order to maintain continuity in topic, this section makes no assumptions about what is known relating to atomicity, and the topic is left for after the current chapter.

### 6.9.3 Atomicity + Independence

This metric is born out of the practical need to know when a pattern is “structurally good enough”. It is meant to replace atomicity and independence, and it has the following advantages:

- It allows the user to determine at a glance if a pattern has satisfactory “linkage structure”. The score has a negative value for patterns which are “incomplete” (with respect to independence), “composite” (with respect to atomicity), fully fixed or completely empty. It can only have a positive value if the pattern has more “self-linkage” than “outwards linkage”.
- Indicates when a pattern is “as small as it gets whilst retaining structure”. This is not the case for **\*-count**, **simplicity**, **atomicity** and **independence**.
- Replaces two metrics with one, so that the resulting Pareto fronts tend to contain fewer items, and the resulting algorithm is also faster.

The metric is defined as follows:

$$\begin{matrix} \text{atomicity} \\ \text{plus} \\ \text{independence} \end{matrix} (x^\psi) = \text{atomicity}(x^\psi) + \text{independence}(x^\psi) \quad (6.12)$$

To note is that **independence** is negative and **atomicity** is positive, which is why the objective has a positive value is both objectives are being maxised at the same time.

#### 6.10 OTHER METRICS

Although most applications of **PSs** will involve the 3 categories of objectives described above, there are others that were developed within this research.

**SAMPLE SIZE** When applying the concept of **PSs** to the domain of Recommender Systems (published in [43]), one aspect arose that was not significant in metaheuristics: the pattern should be fairly common within the  $P_{\text{Ref}}$ .

For this particular context, the interest falls in understanding a **ML** recommender model. It is useful for the stakeholders to understand situations (i.e. patterns) associated with higher performance. These patterns are desirable to stakeholders if they lead to increased performance (already captured by mean fitness and similar metrics) and if they are commonly occurring in the data. This latter objective is defined as:

$$\text{sample\_size}(x^\psi) = |P_{\text{match}}| \quad (6.13)$$

**ROBUSTNESS** Within an optimal solution, there may be a pattern which “ensures high fitness”, whereas the rest has smaller but still positive contributions. In cases where new constraints arise and solutions need to be modified manually (such as the scenarios discussed in [300]), the user is interested in explanations which help to understand which part of the pattern is essential.

The term **robustness** is used to describe a trait within solutions with a relatively good worst-case fitness [300]. This was first discussed in [138], although this relates to classification accuracy in models. The same ideas can be adapted into an objective for **PSs**:

$$\text{robustness}(x^\psi) = \min(F(P_{\text{match}})) \quad (6.14)$$

This metric is to be maximised, so that the lowest fitness possible observed for a pattern is as high as possible.

Consider for example a scenario where each solution represents an investment strategy, and the fitness function represents return-on-investment. The **PS** (\* \* 0 0 \* \* 1 1 1) has a high **mean-fitness** = +140, but this high average is driven by few isolated instances with very high returns, and it actually results in a loss in much of the  $P_{\text{Ref}}$ . Its **robustness** is -20. Consider instead a pattern (\* \* 1 \* \* 0 1 2), which has **mean-fitness** = +30, and **robustness** = +20. This latter pattern is “guaranteed” to bring a positive return, even in its worst case, which makes it more desirable for a scenario where worst-case performance carries a lot of weight.

#### 6.11 ILLUSTRATIVE EXAMPLES ON BENCHMARK PROBLEMS

The many metrics presented in this chapter are better understood by example, and this section will provide scenarios where the differences and motivations behind different objectives are clear. For every problem and combination of objectives, an *approximation* of the Pareto front is obtained and described.

The results shown in this section are created **algorithmically**. Given an optimisation problem, all of its possible **PSs** are created and scored according to a set of objectives. This data is then passed to the Non-Dominated-Sorting algorithm used by NSGA-II in order to find the “true” Pareto front. Any values relating to mean fitness and other objectives are calculated programmatically. The analysis presented here is therefore separated entirely from the **PS** mining process, and instead focuses on just the choice of objectives.

There are problems where it is known what the ideal **PSs** are, which will be called the **target patterns**. There are also problems where such ground truth patterns are subjective. The intended learning outcome of this section is to show that there is no combination of objectives which works well on every optimisation problem.

#### 6.11.1 Motivating mean-fitness, simplicity and atomicity

For a Royal Road problem with 12 bits (3 cliques, each of 4 bits), we know that the target patterns should be (as defined in [198]):

```
(1 1 1 1 * * * * * *)
(* * * * 1 1 1 1 * * * *)
(* * * * * * * * 1 1 1 1)
```

**ONLY USING MEAN-FITNESS** If  $P_{\text{Ref}}$  is the enumerated search space, the target patterns have mean fitness of  $\frac{18}{16}$ . Compare their **mean-fitness** against other patterns (with  $\overline{F}$  indicating **mean-fitness** for brevity)

|                         |                        |
|-------------------------|------------------------|
| 1 1 * * 1 1 * * * * *   | $\overline{F} = 9/16$  |
| 1 1 1 * * * * * * *     | $\overline{F} = 10/16$ |
| 1 * * * 1 * * * 1 * * * | $\overline{F} = 6/16$  |
| 1 1 1 1 1 * * * * * *   | $\overline{F} = 19/16$ |
| 1 1 1 * 1 1 * * * * *   | $\overline{F} = 13/16$ |

If **mean-fitness** is used as the only search metric, the optimal **PS** is a pattern identical to the optimum of the original problem, which is not a satisfactory explanation.

```
(1 1 1 1 1 1 1 1 1 1 1 1)  $\overline{F} = 3$ 
```

**MEAN FITNESS AND \*-COUNT SIMPLICITY** **\*-count simplicity** can be included as an objective to find trade-offs between simplicity and fitness quality:

|                       |                      |                        |
|-----------------------|----------------------|------------------------|
| (* * * * * * * * * *) | <b>*-count</b> = 12, | $\overline{F} = 3/16$  |
| (1 * * * * * * * *)   | <b>*-count</b> = 11, | $\overline{F} = 4/16$  |
| (1 1 * * * * * * *)   | <b>*-count</b> = 10, | $\overline{F} = 6/16$  |
| (1 1 1 1 * * * * * *) | <b>*-count</b> = 8,  | $\overline{F} = 18/16$ |
| (1 1 1 1 1 * * * * *) | <b>*-count</b> = 7,  | $\overline{F} = 19/16$ |
| .....                 |                      | ...                    |
| (1 1 1 1 1 1 1 1 1 1) | <b>*-count</b> = 0,  | $\overline{F} = 48/16$ |

*Note that the 1's were left-aligned here for clarity, and that the real Pareto Front will have the 1's appear in unordered locations since simplicity is not influenced by position.* This combination of objectives leads to a Pareto front of hundreds of patterns, even for small problems. Although the target patterns are within this set, they are effectively indistinguishable from the rest.

**MEAN-FITNESS, ATOMICITY AND \*-COUNT SIMPLICITY** We now introduce atomicity, using linkage information obtained using Perturbation (defined in Section C.4). Which linkage method is being used here is not relevant - the important part is that atomicity is higher when all the variables in a pattern depend on each other in their connection to the fitness function.

The three metrics used in conjunction lead to a large Pareto front. The patterns can be sorted (according to the method shown in Section 6.6.3), and the top PSs are:

```
(1 1 1 1 * * * * * * *)
(* * * * 1 1 1 1 * * * *)
(* * * * * * * * 1 1 1 1)
(1 1 1 1 1 * * * * * *)      worse atomicity, higher mean fitness
(1 1 1 * * * * * * * *)      worse mean fitness, higher simplicity
... ..
```

The target patterns come out on top (which does not happen for other combinations of objectives). Therefore, this combination of objectives is the one recommended for general use cases.

MEAN-FITNESS AND [ATOMICITY + INDEPENDENCE] This final combination of objectives for RoyalRoad is another combination which produces generally desirable results.

The Pareto front is easier to interpret, since it is just the trade-off of 2 metrics.

The patterns in the Pareto front with  $\frac{\text{atomicity}}{\text{plus independence}} > 0$  (here shortened to [A+I]) are the target PSs:

```
(1 1 1 1 * * * * * * *) positive [A+I],  $\overline{F} = 18/16$ 
(* * * * 1 1 1 1 * * * *) positive [A+I],  $\overline{F} = 18/16$ 
(* * * * * * * * 1 1 1 1) positive [A+I],  $\overline{F} = 18/16$ 
```

And every other PS in the Pareto front will have negative [A+I] or a lower mean-fitness. Therefore, this specific combination of metrics is the easiest to use, because useful patterns are clearly identifiable. Unfortunately, it is unknown if for every problem there are patterns with positive [A+I], so general applicability cannot be fully recommended.

### 6.11.2 Knowing when to mine Partial Solutions: OneMax

For a OneMax problem with 12 bits, the target patterns are:

```
(1 * * * * * * * * *)
(* 1 * * * * * * * *)
(* * 1 * * * * * * *)
...
```

In this case, each pattern has \*-count simplicity = 11 and mean-fitness = 6.5. The atomicity of these is null, because each pattern has 1 variable.

If a normal 3-objective PS mining run was carried out, the result would not match the target patterns at all, because those target patterns all score *null* in atomicity so they would not appear in the Pareto front. Instead, the user should first **check if it is even worth using atomicity**, by looking at the linkage table (such as the one shown in Section 5.7).

For this problem, the entire linkage table is composed of zeros, as there is no variable interaction. Knowing this information, the PS mining process is not useful: there is no need to mine for patterns because the user should just investigate the behaviour of each variable independently. This can be achieved by applying mean-fitness to each possible one-variable pattern, and sorting them by mean-fitness.

### 6.11.3 Issues with PS mining: Trapk

For a Trap-k problem of 15 bits (3 cliques, each with 5 bits), the target PSs fall in two categories. The first is similar to what was observed in Royal Road:

$$(1\ 1\ 1\ 1\ 1\ *\ *\ *\ *\ *\ *\ *\ *)\ \overline{F} = 8.375$$

$$(*\ *\ *\ *\ *1\ 1\ 1\ 1\ 1\ *\ *\ *\ *)\ \overline{F} = 8.375$$

$$(*\ *\ *\ *\ *\ *\ *\ *\ *1\ 1\ 1\ 1\ 1)\ \overline{F} = 8.375$$

The second kind of target pattern is:

$$(0\ *\ *\ *\ *\ *\ *\ *\ *\ *\ *\ *)\ \overline{F} = 5.375$$

$$(*\ 0\ *\ *\ *\ *\ *\ *\ *\ *\ *\ *)\ \overline{F} = 5.375$$

$$(*\ *\ 0\ *\ *\ *\ *\ *\ *\ *\ *\ *)\ \overline{F} = 5.375$$

...

The second group might appear incorrect, but actually each of those patterns is useful because they are both extremely simple and still beneficial to fitness *on average*. Compared to patterns such as (0 0 0 0 0 \* \* \* \* \* \* \* \* \*), they are simpler.

This behaviour makes it particularly challenging for PS mining, for two main reasons: (1) univariate patterns are an edge case beyond our scope, and (2) the metaheuristic used for PS mining is not perfect. Trap-k will be used to illustrate these points.

**CAN ONLY FIND CLIQUES** Using **\*-count** simplicity, **mean-fitness** and **atomicity**, the result is a Pareto front which can be easily sorted (as shown for RoyalRoad) which will put the cliques at the top ((11111 \*\*\*\*\* \*\*\*\*\*) for example).

Those single-0 target patterns will be at the bottom, since these have *null* atomicity. The scope of our methodology is to find groups of interacting variables which the user would not be able to find otherwise. The behaviour of individual variables is easily found using traditional summary statistics, and therefore it is considered beyond the scope of our methodology.

#### FAULTS OF THE METAHEURISTIC(S)

Here we discuss how the PS mining process itself has faults rooted in its stochastic implementation.

Consider the atomicity and mean-fitness of the following patterns on a fully enumerated  $P_{\text{Ref}}$ :

$$(1\ 1\ 1\ 1\ 1\ *\ *\ *\ *\ *\ *\ *)\ \text{high atomicity},\ \overline{F} = 8.375$$

$$(0\ 0\ 0\ 0\ 0\ *\ *\ *\ *\ *\ *\ *)\ \text{high atomicity},\ \overline{F} = 7.375$$

We see that the patterns composed of cliques of zeros have just as much atomicity as the true target patterns, and their **mean-fitness** is only slightly lower. The PS mining process will sometimes find these 0-cliques instead of their better counterpart, especially since the “false” cliques are much easier to evolve. Generally, mutating a pattern by swapping a \* for a 0 will increase its **mean-fitness**, where as turning it into a 1 will lower it. In other words, the deceptiveness of Trap-k is inherited by its PS mining task.

The PS mining process is not guaranteed to find all the best PSs in one run, which is why sequential niching (i.e. running the algorithm multiple times) is recommended.

Another issue derives from the *original metaheuristic*, the one which solved the original problem and produced the  $P_{\text{Ref}}$ . Assume that it did not find the global optima, and instead this was the best solution found:

$$(1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1)\quad F = 14$$

Then, the  $P_{\text{Ref}}$  will be biased such that the middle clique has the value (1 1 1 1 1) very rarely and (0 0 0 0 0) will be overrepresented. Therefore, the pattern (\* \* \* \* 0 0 0 0 0 \* \* \* \*) might have a mean fitness higher than (\* \* \* \* 1 1 1 1 1 \* \* \* \*) because the former is associated with highly optimised solutions and the latter might have existed in early generations.

The end result of this faulty  $P_{\text{Ref}}$  is:

|                                 |                 |                              |
|---------------------------------|-----------------|------------------------------|
| (1 1 1 1 1 * * * * * * * * *)   | high atomicity, | $\overline{F} = \text{high}$ |
| (* * * * * 0 0 0 0 0 * * * * *) | high atomicity, | $\overline{F} = \text{high}$ |
| (* * * * * * * * * 1 1 1 1 1)   | high atomicity, | $\overline{F} = \text{high}$ |

The **PS** mining process can only succeed if the  $P_{\text{Ref}}$  contains a sample of solutions relevant to the user's needs. Ensuring that the original metaheuristic finds desirable solutions is beyond the scope of this thesis, although our methodology can help detect faults within a sample of solutions.

## 6.12 POST-PROCESSING VIA DESCRIPTORS

By themselves, **PSs** are intended to be understandable because they are simpler than normal solutions, but their presence does not always explain *why* their presence is associated with improved fitness. To resolve this, the concept of **descriptors** is introduced.

The following terminology is introduced here:

- **Proxy-fitness function:** a function which is part of - or otherwise related to - the fitness function. It is a function which maps solutions to a single scalar, similarly to the fitness function. There may be many proxy-fitness functions.
- **Descriptor:** When a **PS** is presented to the user, the system is able to identify if the pattern appears to be “optimising” any of the proxy fitness functions, either maximising or minimising. The pattern has an average value for every proxy-fitness function, and if this appears to be the result of maximisation or minimisation, the proxy-fitness function and its average are the **descriptor**.

Descriptors allow the researcher to introduce domain knowledge into the **PS** explanation system. The user of an optimisation problem may not be able to understand the full fitness function, but they may have heuristics and “close-enough” metrics that they know (or suspect) to be relevant to the full fitness function.

The concept of descriptors is designed to be aligned with the main fundamentals of this research project:

- it is implemented from a **static dataset**, and does not require fitness function or proxy-fitness function evaluations for solutions beyond the original set in the  $P_{\text{Ref}}$ ;
- it allows for **flexibility**: the researcher can implement any proxy-fitness functions they see fit, in order to capture the explainability motivations they are interested in;
- it **does not promise to find causal relationship**, and provides an insightful correlation, which tells the user what are the main factors which determine the fitness function.

Note that the proxy-function is the raw information put into the system, and the **descriptors** is the set of proxy-fitness functions which are relevant because they have extreme values.

It is important to note that descriptors are associations rather than causal relations. This is the main motivation behind using the term “descriptor”: it is a description of an observed association.

### 6.12.1 Descriptors example

Consider a **BT** example scenario where our system was applied, and a **PS** was found. The pattern appears to have high *mean-fitness*, but it is not clear why. The system then reports 2 descriptors: **variance of electrical engineers Mondays**, and **variance of network engineers on Fridays** have very low averages. These are not direct causes as to why the pattern is beneficial to fitness, but it gives an indication to the human user in what properties they might want to investigate and verify for themselves. Mathematically verifying if having low variance in a certain aspect will maximise the objective is a hard task, but the human user may already have an intuition for this, and thus the descriptor aligns with their domain knowledge and proves useful.

The researcher involved in this system would have programmed many possible properties that are likely to relate to the overall fitness, perhaps for every skill and every week. Each of these is a proxy-fitness function, and when looking at a **PS** only the “maximised” or “minimised” ones are shown, at which point they take the name of *descriptor*.

The definition of descriptors presented here is the same as what is presented in [46], but in greater detail in order to give a better idea of its design and implementation.

Consider the example shown in Section 5.7 which uses a multi-dimensional Knapsack problem. In this problem, the fitness function is essentially a sum of values (the calories for each item), but it has various constraints.

A user trying to understand the outcome of this optimisation problem is likely to use the “unconstrained” fitness function in their understanding, and therefore there is merit in using this for explanations.

### 6.12.2 Formalising the concept of “relatively high / low”

The aim of descriptors is to determine if the average proxy-fitness value for a **PS** is “abnormally high” or “abnormally low” for a certain pattern, which may appear to be a simple task on the surface. For this example, we will use a reference population as shown in Table 6.2, with a single proxy-fitness column.

Table 6.2: Example Reference Population table, with proxy-fitness column

| Solution  | Fitness | Proxy-fitness |
|-----------|---------|---------------|
| (1 2 1 3) | 5.2     | 3.1           |
| (2 2 0 0) | 1.5     | 3.0           |
| (1 2 3 4) | 3.2     | 3.1           |
| (1 1 3 0) | 7.9     | 2.6           |
| ...       | ...     | ...           |

Consider the pattern (\* 2 \* \*), which matches a set of solutions. The average value of the proxy-fitness functions for these solutions might be 3.1, and placing it within the distribution of the proxy-fitness column we may find that it is at the 2.5% percentile. This can be reported to the user, perhaps based on it being above a certain threshold percentile.

The above example is a situation where it is easy to determine if the proxy-fitness value has a relatively extreme value. Now consider a different reference population, with a different descriptor in Table 6.3.

Table 6.3: Example Reference Population table, with descriptor column

| Solution  | Fitness | proxy-fitness |
|-----------|---------|---------------|
| (1 2 1 3) | 5.2     | 11            |
| (2 2 0 0) | 1.5     | 11            |
| (1 2 3 4) | 3.2     | 13            |
| (1 1 3 0) | 7.9     | 15            |
| (3 1 0 3) | 7.9     | 13            |
| ...       | ...     | ...           |

In this case, the descriptor only takes a small quantity of discrete values, perhaps where many have the value 11. Perhaps we find that for the pattern  $(* 1 * *)$  the average descriptor value is 11.2. Is this value low, or is it fairly common for a pattern to have a low average value for the descriptor? In other words, does this pattern have an unusually low average descriptor value, *compared to other patterns*?

This question can be taken literally and implemented empirically. A set of randomly generated patterns can be generated, and for each the average proxy-fitness can be found. This produces a distribution, which even for a proxy-fitness function that is not “smooth” will have a smooth distribution, as shown in Figure 6.1.

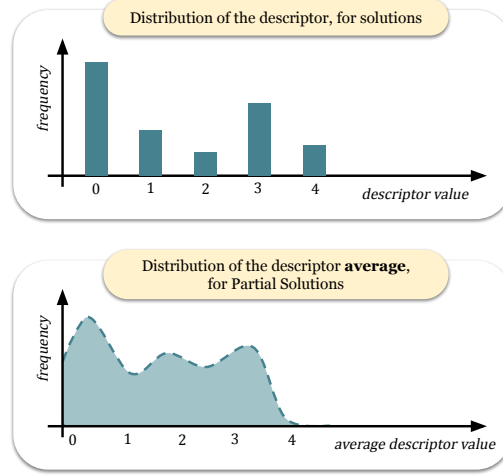


Figure 6.1: Distribution for a proxy-fitness function that is discrete, and the corresponding distribution for its average (over the space of Partial Solutions).

This new distribution can now be used to determine if a pattern’s average descriptor value position within the distribution. Going back to the example, we may say that the value of 11.2 for the descriptor is high, because of 1000 randomly generated patterns it ranks in the top 1% for average descriptor value.

Before getting into the formal definition, there is an additional consideration. For the pattern  $(* 1 * *)$  we may find that its descriptor is in the top 50% for its average descriptor value, but that is when comparing it against any possible randomly generated pattern. When compared against other patterns with a single fixed value, that average might actually be extremely high. In other words, the size of a **PS** is an important consideration when determining if a proxy-fitness value is unusual or not. Therefore, when determining if a pattern’s average descriptor value is high or low, it should be compared against a sample of patterns that have the same quantity of fixed variables.

### 6.12.3 Mathematical definition

Finally, the formal definition of a descriptor can be defined. Here, it is defined for a single proxy-fitness function but it could be defined for many at the same time.

In the context of a  $P_{\text{Ref}}$  and a pattern  $x^\psi$ , we can find the average value of a proxy-fitness function (Equation (6.15)).

$$\text{avg\_proxy}(x^\psi, \text{proxy}) = \text{avg}(\{\text{proxy}(s) \text{ s.t. } s \in P_{\text{match}}\}) \quad (6.15)$$

Then, we generate a set of random patterns, which have the same quantity of fixed values as  $x^\psi$ . Assume that there is a function called `random_PS(size = a)` which randomly generates a pattern with order  $a$ . This is used to create a set of patterns which has the same fixed-variable count as  $x^\psi$  (Equation (6.16)).

$$\begin{aligned} \text{sampled\_PSs}(x^\psi) &= \{y_1^\psi, \dots, y_{\text{ps\_samples}}^\psi\} \\ \text{where } y_i^\psi &= \text{random\_PS}(\text{size} = \text{fixed\_count}(x^\psi)) \end{aligned} \quad (6.16)$$

Then, it is time to see where  $\text{avg\_descriptor}(x^\psi)$  ranks amongst the values of  $\text{avg\_proxy}(\text{sampled\_PSs}(x^\psi))$ . The Empirical Cumulative Distribution Function (**ECDF**) will be used, where  $\text{ECDF}(a, B)$  returns the percentile of the value  $a$  within the collection of values  $B$  (so that the result is 0 when it is the lowest, 1 when it is the highest, 0.5 when it is the median).

$$\text{percentile}(x^\psi, \text{proxy}) = \text{ECDF}(\text{avg\_proxy}(x^\psi, \text{proxy}), \text{avg\_proxy}(\text{sampled\_PSs}(x^\psi))) \quad (6.17)$$

Finally, there may be many proxy-fitness functions, and only the significantly high or low values should be shown. A threshold (**descriptor\_threshold**) will determine how high or low **avg\_proxy** needs to be for it to be shown. When **descriptor\_threshold** = 0.1, then the percentile needs to be lower than 0.1 (top 10%) or higher than 0.9 (bottom 10%). The function that determines whether a descriptor should be shown is in Equation (6.18).

$$\text{should\_show}(\text{perc}) = \begin{cases} \text{true} & \text{if } \text{perc} > \text{descriptor\_threshold} \\ \text{true} & \text{if } \text{perc} < 1 - \text{descriptor\_threshold} \\ \text{false} & \text{otherwise} \end{cases} \quad (6.18)$$

In summary, for a given pattern  $x^\psi$  and a proxy-fitness function **proxy**, the user will be shown  $\text{avg\_proxy}(x^\psi, \text{proxy})$ , which has percentile  $= \text{percentile}(x^\psi, \text{proxy})$ , but only if **should\_show**(percentile) is true.

#### 6.12.4 Design considerations behind descriptors

There are a few possible objections that the reader might have relating to descriptors:

1. **Why use this complicated process** of generating random patterns and generating a distribution of *averages*? Why not use the original distribution of descriptor values?
2. Why is the **ECDF** used when a **statistical test** would be more mathematically sound? In that case, the resulting significance of a descriptor would be the p-value of a test.

For the first point, the issue stems from discrete or discontinuous proxy-functions, which are not uncommon (such as binary constraints). Determining where a value lies within such distributions is surprisingly arbitrary and leads to numbers with little meaning to the end-user.

For example, consider a population where 300 solutions have a proxy-fitness equal to 1, and 700 has a proxy-fitness equal to 0. If a pattern matches solutions such that the average proxy-fitness is 0.6 on average, it is hard to determine how *relatively* high this value is. Instead, one could generate many patterns, and see what averages are observed. This forms a smooth distribution, where 0.6 can be given a percentile.

By comparing against a sample of randomly generated patterns, the resulting percentiles are much more stable whilst still being understandable to the user.

As for the second point, the issue is not that we are trying to determine *if* a descriptor value is higher or lower than a midpoint or another arbitrary point, but rather we want to measure *how extreme* they are. Perhaps the pattern  $(* * * 1)$  matches solutions with proxy-fitness values of values  $\{3, 4, 4, 5, \dots\}$  and we find through a Mann-Whitney-U test (non parametric two tailed independent samples to compare averages) that the average of these values is higher than the population average, with a p-value much lower than 0.05. Unfortunately this does not indicate how *extreme* these values are, and just measures if they are in the greater half of the distribution.

#### 6.12.5 Expected usage for descriptors

Descriptors aim to allow the user to understand why a **PS** might be critical to good fitness in solutions. The calculations used to determine whether a descriptor is noteworthy (by being having a sufficiently high or low average proxy-fitness value, based on a user-set threshold) can be cached and are efficient, therefore they are not a concern in terms of execution time.

In general, the user is incentivised to register many proxy-fitness functions, with these being as specific as desired. Since there is a mechanism to automatically ignore values which are not noteworthy, adding many descriptors will not overwhelm the user who is viewing the results (unless there are a lot of relevant descriptors, which might be a necessary evil).

This method was introduced in [46] for the Staff Rostering problem. The proxy-fitness functions used there could be as specific as “The total quantity of workers allocated on Wednesdays which have the skill **woodworking**”. For the Staff Rostering problem, the fitness function is a sum of many smaller functions, and each of these was translated into a proxy-fitness function.

### 6.13 CLUSTERING OF PATTERNS

This step is optional and applies to scenarios where there are too many patterns to be shown to the user. This is often an inevitable issue of using multi-objective methods.

The examples seen in Section 6.11 showed scenarios in which the Pareto front of **PSs** is large, beyond what a user is reasonably willing to sift through. Although the right combination of objectives can reduce the quantity of patterns found, it might be the case that having lots of patterns is inevitable.

In these cases, the user might be easily overwhelmed and would rather see “patterns within the patterns”. This can quickly become confusing, so instead a simpler step is taken: **cluster the patterns by their similarity in decision space**, and present the **centroids** to the user. Clustering is done in decision space, as the interest lies in finding groups of “visually” similar patterns.

There are plenty of heuristics to implement this approach, and traditional clustering approaches will usually require the user to manually select the quantity of clusters to find. Since the user cannot know in advance how many **PSs** they should expect, in this work we opted for a parameter-less method: HDBSCAN [41]. This was used instead of traditional DBSCAN due to its ability to detect clusters of differing densities, and therefore makes fewer assumptions.

In order to apply clustering, a **distance metric** has to be defined over patterns. Although one was defined for niching in [101], that metric was developed for algorithmic search performance rather than “visual similarity”. Instead, we propose to use what is essentially a **Hamming distance**: the quantity of different variables between two patterns. Although this would work, it is tuned here slightly so that the “difference” between values is as follows:

|       |    |               |                  |
|-------|----|---------------|------------------|
| *     | vs | fixed value : | difference = 1   |
| fixed | vs | fixed value : | difference = 0.9 |

In order to understand that 0.9, one should consider the following:

which pattern is more similar to (1 1 \* \*) : (\* \* 0 0) or (0 0 \* \*)?

Visually, the second pattern is more similar, because it has the (\*)’s in the same positions. Using 0.9 instead of 1 is used to encode a slight bias towards [same-position but different value] differences.

Clustering using HDBSCAN has a feature which is both an advantage and a disadvantage: it categorises certain patterns as noise because they are too dissimilar from the rest of the Pareto front. This might be both an effective way of improving user experience, but might also prevent useful patterns from being shown to the user.

### 6.14 EXECUTION TIME

One of the main objectives considered throughout this research is to make the generation of explanations fast enough in order to reduce barriers to usage and to allow the system to be used interactively.

To determine the execution speed of **PS** mining, this section will provide an evaluation which applies **PS** mining on NK-problems (defined in Section 2.2.6) with differing values for  $N$  (representing the quantity of variables) and  $K$  (different levels of epistasis, i.e., linkage). The evaluation is also applied on  $P_{\text{Ref}}$  of different sizes, and different **PS** objectives. These parameters are modified in order to verify how problem size, problem linkage, sample size and objectives combination affect the execution time of the **PS** mining process. The **PS** mining budget is kept constant to 10000, so that we can get an estimation of runtime per 10000 evaluations.

The experimental procedure will test every combination of the following parameters:

- $N \in \{20, 50, 100\}$  is the quantity of variables in the NK-problem.
- $K \in \{2, 5, 10\}$  is the size of each “interaction clique” in the NK problem. Larger  $K$  will lead to larger **PS**s.
- $\text{Pref\_size} \in \{5000, 10000, 15000\}$  is the evaluation budget for the metaheuristic solving the NK-problem. This is equal to the quantity of solutions in the resulting  $P_{\text{Ref}}$ .
- $\text{objectives} \in \{\text{SM}, \text{SMA}, \text{M(A+I)}\}$  represents the possible combinations of objectives in the **PS** mining process. The letter **S** represents **\*-count simplicity**, **M** represents **mean fitness**, **A** for **atomicity**, and **(A+I)** stands for the **A+I** metric. The combinations of objectives are chosen as they are intended as the most typical use cases for our system. The atomicity and independence metrics make use of linkage information found via dominance-based linkage, since it is the best linkage method which is feasibly executable for a problem with 100 variables, as seen in Section 7.5.

For every combination of these parameters, 30 runs are executed and timed. Each run consists of the following steps:

- An NK-problem instance with parameters  $N$  and  $K$  is generated *randomly*.
- The NK problem instance is solved by a standard **GA** ( $\mu + \lambda$  algorithm, population size = 100, mutation rate =  $1/L$ , crossover rate = 0.5, uniform mutation and simulated binary crossover) with an evaluation budget of **Pref\_size**.
- The solutions evaluated by the metaheuristic are stored in a  $P_{\text{Ref}}$ , which will contain **Pref\_size** solutions.
- A **PS** mining process is applied on the  $P_{\text{Ref}}$ , with an evaluation budget of 10000 and all the **Any-Pattern** operators described in Section 6.5.2. The objectives used are those determined by **objectives**.
- The **PS** run is executed and timed. The runtime includes the time required to obtain the linkage table (if necessary), although this was found to be negligible (<5% of total runtime).

The results of these runs are presented in Table 6.6, Table 6.5 and Table 6.7, but they are summarised as follows: runs take between 4 and 5.5 seconds generally, and the runtime appears to be determined by  $N$  and the **PS** mining objectives. A table of results is shown in Table 6.4.

A more thorough evaluation is provided below.

| <b>N</b> | <b>Objectives</b> |                   |                 |
|----------|-------------------|-------------------|-----------------|
|          | <b>SM</b>         | <b>SMA</b>        | <b>M(A+I)</b>   |
| 20       | 4.120 $\pm$ 0.17  | 4.185 $\pm$ 0.18  | 4.17 $\pm$ 0.12 |
| 50       | 4.490 $\pm$ 0.17  | 4.586 $\pm$ 0.11  | 4.69 $\pm$ 0.26 |
| 100      | 5.096 $\pm$ 0.15  | 5.2463 $\pm$ 0.17 | 5.51 $\pm$ 0.22 |

Table 6.4: Average runtime ( $\pm$  standard deviation) for each combination of  $N$  and **PS** mining objectives tested, when the **PS** evaluation budget = 10000.

DETERMINING WHICH PARAMETERS INFLUENCE RUNTIME We determine how strongly each of the numeric parameters correlates to runtime, by using Spearman  $\rho$ , which measures the strength of the *monotonic* relationship. We present the correlation and the p-value of the test, with significance set to 0.05:

- N has  $\rho = 0.918$  (strong correlation), p-value  $\ll 0.001$  (highly significant)
- K has  $\rho = -0.0357$  (no relationship), p-value = 0.083 (not significant)
- Pref\_size: 0.03 (moderate), p-value = 0.07 (not significant)

To determine the importance of **objectives**, we run a One-way ANOVA test to determine if each objective category has a different mean. The result of this test has p-value  $\ll 0.001$  (effect size  $F = 40.1$ ), meaning that there is strong evidence suggesting that the choice of objectives influences the runtime. The average runtime for each combination of N and **objectives** is reported in Table 6.4.

These results suggest that the **PS** mining process is fast in most scenarios, taking between 4 and 5.5 seconds for every 10000 **PS** evaluations, with the execution time being determined mainly by the quantity of variables in the solutions. Factors such as the size of the  $P_{\text{Ref}}$  or the degree of epistasis do not affect runtime significantly.

| Objectives = SMA |    |                           |                   |                   |
|------------------|----|---------------------------|-------------------|-------------------|
| N                | K  | Average runtime $\pm$ std |                   |                   |
|                  |    | Pref_size = 5000          | Pref_size = 10000 | Pref_size = 15000 |
| 20               | 2  | 2.12 $\pm$ 2              | 4.22 $\pm$ 2      | 6.21 $\pm$ 2      |
| 20               | 5  | 2.08 $\pm$ 2              | 4.16 $\pm$ 2      | 6.29 $\pm$ 2      |
| 20               | 10 | 2.08 $\pm$ 2              | 4.20 $\pm$ 2      | 6.23 $\pm$ 2      |
| 50               | 2  | 2.25 $\pm$ 2              | 4.53 $\pm$ 2      | 6.88 $\pm$ 2      |
| 50               | 5  | 2.25 $\pm$ 2              | 4.52 $\pm$ 2      | 6.83 $\pm$ 2      |
| 50               | 10 | 2.25 $\pm$ 2              | 4.52 $\pm$ 2      | 6.80 $\pm$ 2      |
| 100              | 2  | 2.46 $\pm$ 2              | 4.96 $\pm$ 2      | 7.49 $\pm$ 2      |
| 100              | 5  | 2.45 $\pm$ 2              | 4.96 $\pm$ 2      | 7.46 $\pm$ 2      |
| 100              | 10 | 2.45 $\pm$ 2              | 4.94 $\pm$ 2      | 7.46 $\pm$ 2      |

Table 6.5: Average runtime ( $\pm$  standard deviation) for Objectives = SMA, across combinations of N, K, and Pref\_size.

| Objectives = SM |    |                           |                   |                   |
|-----------------|----|---------------------------|-------------------|-------------------|
| N               | K  | Average runtime $\pm$ std |                   |                   |
|                 |    | Pref_size = 5000          | Pref_size = 10000 | Pref_size = 15000 |
| 20              | 2  | 2.03 $\pm$ 2              | 4.20 $\pm$ 2      | 6.11 $\pm$ 2      |
| 20              | 5  | 2.03 $\pm$ 2              | 4.12 $\pm$ 2      | 6.12 $\pm$ 2      |
| 20              | 10 | 2.04 $\pm$ 2              | 4.07 $\pm$ 2      | 6.20 $\pm$ 2      |
| 50              | 2  | 2.18 $\pm$ 2              | 4.44 $\pm$ 2      | 6.68 $\pm$ 2      |
| 50              | 5  | 2.18 $\pm$ 2              | 4.41 $\pm$ 2      | 6.66 $\pm$ 2      |
| 50              | 10 | 2.18 $\pm$ 2              | 4.39 $\pm$ 2      | 6.61 $\pm$ 2      |
| 100             | 2  | 2.37 $\pm$ 2              | 4.84 $\pm$ 2      | 7.28 $\pm$ 2      |
| 100             | 5  | 2.37 $\pm$ 2              | 4.85 $\pm$ 2      | 7.30 $\pm$ 2      |
| 100             | 10 | 2.36 $\pm$ 2              | 4.79 $\pm$ 2      | 7.27 $\pm$ 2      |

Table 6.6: Average runtime ( $\pm$  standard deviation) for Objectives = SM, across combinations of N, K, and Pref\_size.

| Objectives = M(A+I) |    |                           |                   |                   |
|---------------------|----|---------------------------|-------------------|-------------------|
| N                   | K  | Average runtime $\pm$ std |                   |                   |
|                     |    | Pref_size = 5000          | Pref_size = 10000 | Pref_size = 15000 |
| 20                  | 2  | 2.11 $\pm$ 2              | 4.27 $\pm$ 2      | 6.41 $\pm$ 2      |
| 20                  | 5  | 2.15 $\pm$ 2              | 4.22 $\pm$ 2      | 6.38 $\pm$ 2      |
| 20                  | 10 | 2.12 $\pm$ 2              | 4.23 $\pm$ 2      | 6.30 $\pm$ 2      |
| 50                  | 2  | 2.30 $\pm$ 2              | 4.71 $\pm$ 2      | 7.16 $\pm$ 2      |
| 50                  | 5  | 2.30 $\pm$ 2              | 4.68 $\pm$ 2      | 7.01 $\pm$ 2      |
| 50                  | 10 | 2.31 $\pm$ 2              | 4.68 $\pm$ 2      | 6.87 $\pm$ 2      |
| 100                 | 2  | 2.59 $\pm$ 2              | 5.54 $\pm$ 2      | 8.39 $\pm$ 2      |
| 100                 | 5  | 2.63 $\pm$ 2              | 5.41 $\pm$ 2      | 8.28 $\pm$ 2      |
| 100                 | 10 | 2.57 $\pm$ 2              | 5.30 $\pm$ 2      | 8.02 $\pm$ 2      |

Table 6.7: Average runtime ( $\pm$  standard deviation) for Objectives = M(A+I), across combinations of N, K, and Pref\_size.

#### 6.15 EFFECT OF REFERENCE POPULATION ON MINING PROCESS

The PS Mining process rooted by the  $P_{\text{Ref}}$ , and thus it is worth investigating if and how this relationship affects the results. One particular concern is that if the metaheuristic converged too quickly, then the  $P_{\text{Ref}}$  will be too heavily biased towards positive solutions, and the PS mining process will be unable to distinguish truly positive patterns from *slightly* positive or even hitch-hiking effect. One way this would be mitigated is by introducing solutions which are uniformly randomly generated into the  $P_{\text{Ref}}$ .

This was evaluated by running the same PS mining pipeline, but starting from different  $P_{\text{Ref}}$ . In both cases, a Simulated Annealing algorithm (effectively iRoster, uniform mutation rate of  $\frac{1}{L}$  was used with the full budget) is used. In one case, the solutions evaluated form this form the entire  $P_{\text{Ref}}$ . In the second case, they only form 50% of it, as the other is composed of uniformly randomly generated solutions.

The evaluation is applied on two benchmark problems (Staff Rostering Royal Road and Staff Rostering Graph Colouring) (the same ones used in Table 6.1). The evaluation will measure how often a run will contain *all* of the target PSs (as determined by the design of the benchmark problems, discussed in Table 6.1), and this serves as our “success” criterion. Each possible configuration was ran 100 times, and the proportion of successful runs is reported.

The evaluation uses 3 possible configurations of our pipeline: a low-budget many-run configuration (population size is 50, budget-per-run in 1000, 10 runs in total), a medium option (population size is 1000, budget is 5000, 2 runs in total) and a single-shot option (population is 200, evaluation budget is 10000, 1 run).

Table 6.8: Average proportion of discovered groups for the SR-RR and SR-GC problems, applied on differing reference populations (Best performing for each column is shown in **BOLD**)

| Multi-Objective GA | Run settings |              | Ref. Pop = 100% SA |            | Ref. Pop = 50% uniform, 50% SA |            |
|--------------------|--------------|--------------|--------------------|------------|--------------------------------|------------|
|                    | Pop. Size    | Eval. Budget | SR-RR              | SR-GC      | SR-RR                          | SR-GC      |
| NSGA-II            | 50           | 1000         | <b>67%</b>         | <b>43%</b> | <b>93%</b>                     | <b>68%</b> |
|                    | 100          | 5000         | 33%                | 18%        | 86%                            | 27%        |
|                    | 200          | 10000        | 22%                | 13%        | 77%                            | 18%        |

The results shown in Table 6.8 indicate that configurations where the  $P_{\text{Ref}}$  was mixed with uniformly random solutions (the 50% Simulated Annealing, 50% uniformly random) had a higher frequency of successful runs in all cases. This indicates a potential weakness of the PS Mining process: the sample of the search space provided by the metaheuristic needs to be *diverse* enough in order to accurately find PSs. Although this is a potentially serious issue, it should be noted that the Simulated Annealing algorithm used here is

a strongly converging algorithm, and thus the effect is likely to be less pronounced in other cases. There is a simple fix for this issue if encountered: simply to extend the  $P_{\text{Ref}}$  with randomly generated solutions. Because the **PS** Mining process does not depend on the solutions coming from one source, this can be done trivially.

Linkage learning as a topic is relevant to this research for its explainability potential: knowing which variables are linked can help the user in understanding how the problem can be decomposed into smaller parts.

The process of computing linkage information is necessary in this thesis to implement the **atomicity** search metric for **PSs**. This chapter will give an overview of linkage learning and provide the definition of many formulas used in Section 6.9.

There are many definitions for linkage, which are relevant for different contexts and make different assumptions about the data being analysed. Although linkage detection might appear to be a solved problem, existing methods were found to be insufficient for our purposes. This chapter explores the issue of linkage in depth:

- Section 7.1: An overview of the multiple **meanings of linkage**.
- Section 7.2: A discussion of various **categorisations** for linkage detection methods. Implementations for the linkage methods themselves are provided in Chapter C
- Section 7.4: Description of two **novel linkage detection methods**: Mutual Information of Survival (Section 7.4.1) and Dominance-based linkage (Section 7.4.2).
- Section 7.5: The definition of a novel **linkage evaluation** method in metaheuristic contexts, so that linkage detection methods can be compared. Experimental results are provided in Section 7.5.6.

The experiments show that most methods in the literature either perform poorly on benchmark problems, or they perform well but scale poorly in terms of time complexity. The novel linkage method of Dominance-based linkage provides a better trade-off between speed and quality of results.

## 7.1 INTRODUCTION

There are several definitions of linkage from various domains:

- In **genetics**, linkage represents how certain groups of genes in DNA have a “greater than average chance of being inherited from the same parent”, also called “epistasis” [59].
- In **statistics**, linkage can mean **correlation**, where the values observed for one variable are in some way dependent or influenced by another variable.
- In **statistics**, linkage can mean that the probability distribution of a categorical variable is dependent on one or more of the other categorical variables. This is called **conditional probability distributions**, where **information theory** and **Bayesian theory** are often used to detect linkage [218, 35].
- In **machine learning**, linkage also means correlation [115], but especially relevant for **feature selection**: certain variables should not be included as inputs for the model because they bring the same information as already-included variables. Therefore, linkage is defined as two variables which, when used in conjunction, provide less information (therefore worse predictive performance) than the sum of their parts [157]
- In **machine learning** and **EDAs**, linkage can mean the importance of a set of variables in combination. Similarly to how variable importance can be defined for a single variable, linkage may also be defined for multiple variables [282]. This includes the coefficients of models which are multi-variate (such as the Markov models of Brownlee et al [37, 36]). A group of variables may be more *important* than the sum of its parts.

- In **metaheuristics**, linkage can mean that certain **crossover** operators will display an effect similar to real-world genetics, where certain genes tend to be inherited from the same parent [59].
- In **metaheuristics**, linkage can mean that a variable's effect on fitness is dependent on other variables [59]. The fitness function might be thought of as a hierarchy of sets of variables which interact as cliques.

These are different points of view of a phenomenon which takes place across many fields. The same ideas are recurring, but there are also significant differences, which are discussed throughout this chapter. This chapter's point of view is from metaheuristics, but the discussions provide insights relating to the other fields mentioned.

### 7.1.1 The metaheuristic context

This chapter focuses on the following use case: a **metaheuristic** is used to solve a **combinational optimisation problem**, and any evaluated solutions are stored (the  $P_{\text{Ref}}$ ). Linkage is computed from the  $P_{\text{Ref}}$  and used in two ways:

- a **linkage table**, showing how strongly linked any pair of variables is. This is for explainability, to have a better idea of what groups of variables are dependent on others, and to inform how to manually modify solutions
- an **atomicity metric**, used to search for **PSs**.

In the following, “fitness” refers to the quality of solutions, but in general it can be understood to mean a value indicating the quality of each item of a dataset.

### 7.1.2 Pairwise linkage for computational speed

Linkage is often **computationally expensive** to calculate because it involves analysing relationships between many combinations of values. Therefore, the most common kind of linkage is **pairwise** (also called bivariate), where the measurement is between *pairs* of variables. The outcome of pairwise linkage detection can be thought of as an  $L \times L$  matrix, where  $L$ . This table is relatively small, and can be calculated at once and used to approximate higher-order linkages. This aspect is a crucial detail for the implementation of **atomicity** as a search metric for **PSs**.

For the rest of this chapter, we will focus on bivariate linkage.

## 7.2 CHARACTERISATIONS OF LINKAGE

This section will present various ways in which one can characterise linkage learning methods.

### 7.2.1 Fitness-informed vs fitness-independent

This terminology is not present in the literature, and it is introduced here to highlight how many linkage learning methods entirely ignore the fitness function. Section 7.2.1.1 is an example of a common scenario where this terminology prevents confusion.

This categorisation is based on whether fitness values are used or not:

1. **Fitness-independent methods:** we check if the variables affect each other's values within a given dataset. Fitness values are absent or ignored.
2. **Fitness-informed methods:** fitness is determined by the values of the variables, and we determine when this relationship depends on their *combination* of values rather than their independent values.

The difference, shown in Figure 7.1, is subtle but significant.

An example of **fitness-independent** linkage detection is “people who take omega-3 supplements are more likely to take vitamin-D supplements than those who do not”.

In the case of a “**fitness-informed**” linkage calculation, the variable’s distribution does not matter, but rather their relationship to the fitness values. For example, “[taking omega-3 supplements] and [taking vitamin-D supplements] and [regular exercise] reduces the risk of cancer by 61%, which is much more than the reduction from each of those practices in isolation ” [25].

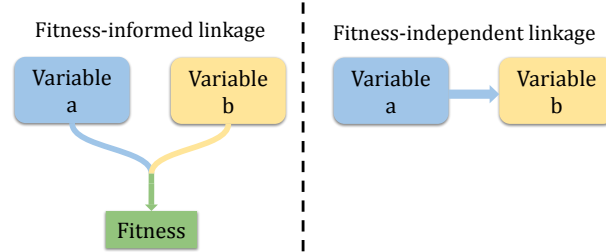


Figure 7.1: The two different main kinds of linkage learning. In “fitness-informed” linkage, each variable affects fitness, but the effect of one variable is dependent on another. In “fitness-independent” linkage detection, it is the values that are affected: the value of a variable is (at least partially) determined by the value of another.

This separation is relevant because the mathematical approaches for these groups are not necessarily equivalent. In particular, the first category (fitness-independent) is more suitable to analyse the distribution of the solutions in the given data, and therefore the **behaviour of the metaheuristic**. If single-point crossover is used, for example, there might be conditional-probability-linkage between adjacent variables [117]. The fitness function influences the behaviour of metaheuristics, and thus the linkage information for these methods is an *indirect* description of the fitness function.

The second category (fitness-informed) is more suitable for analysing the **fitness function directly**, and the utility of the given set of solutions is simply to provide sufficient samples.

#### 7.2.1.1 Practical example

The distinction introduced above is not generally made in the literature, which leads to confusion about what is meant when the term “linkage” is used. Consider a **feature selection** scenario, where two variables are said to be “linked”, then both of these statements could be valid:

1. one of them should be removed, because there is *redundant information*. Knowing one variable leads to already knowing the value of the other,
2. it is particularly important to keep both of them, because their combination is what influences the outcome.

The first case falls in the fitness-independent linkage detection, whereas the second case falls in the fitness-informed category.

#### 7.2.2 Paired and unpaired samples approaches

This categorisation is introduced here to distinguish two main ways of implementing linkage, one of which suffers particularly from biases in the sample data. Metaheuristics are particularly prone to this bias (by design, since they converge), and the example in Section 7.2.2.1 shows how much the distinction introduced here can make a difference.

Another binary classification can be made for linkage, which is similar to how t-tests can be carried out: **unpaired samples** (also called independent samples) and **paired samples** measurements. Considering a set of variables  $V$ , we want to measure how strongly linked they are to each other. The two possible approaches are:

- **unpaired sampling:** Information relating to variables is **globally** measured. Groups of solutions are analysed in bulk, such as by checking their average fitness.
- **paired sampling:** solutions are modified slightly only in variables within  $V$  to observe changes in fitness caused by those variables. The other variables are left unchanged, so that outside influences (outside of  $V$ ) are controlled.

In terms of **quality of results**, paired sampling is more reliable because unpaired samples can produce more false positives if the input data is biased. This is best explained by example, which will be given shortly in Section 7.2.2.1.

In terms of **computational complexity**, paired sampling is often infeasible, since they rely on finding (or generating and evaluating) solutions which differ only on the variables of interest. Therefore, the two approaches fulfil different trade-offs of efficiency and accuracy.

#### 7.2.2.1 Example comparing unpaired samples and paired samples

Consider a situation where we intend to find the linkage between binary variables  $u$  and  $v$ , and we use an **unpaired samples method**. The full problem has three variables ( $u$ ,  $v$  and  $w$ ), and we have evaluated a *subset* of the search space. The fully evaluated search space and its sample are presented in Table 7.1.

| u | v | w | F  | In sample? |
|---|---|---|----|------------|
| 0 | 0 | 0 | 0  | No         |
| 0 | 0 | 1 | 30 | Yes        |
| 0 | 1 | 0 | 12 | Yes        |
| 0 | 1 | 1 | 42 | Yes        |
| 1 | 0 | 0 | 9  | Yes        |
| 1 | 0 | 1 | 39 | Yes        |
| 1 | 1 | 0 | 21 | No         |
| 1 | 1 | 1 | 51 | Yes        |

Table 7.1: Example of a scenario where an optimisation problem with three variables ( $u$ ,  $v$  and  $w$ ) and a fitness function  $F$  is investigated. All the solutions in the search space are shown, but the linkage learning method only has access to a sub-sample of them (rightmost column).

**UNPAIRED SAMPLES APPROACH** To measure the linkage between  $u$  and  $v$ , we use averages on the samples available. The overall average fitness is 30.5, when  $u = 1$  the average is 33, when  $v = 1$  we have an average of 27, and finally when  $(u, v) = 1$  we get an average of 51. These results would suggest the presence of strong linkage between the two variables, since setting both variables to 1 results in an average fitness much higher than expected if they were independent.

In reality, this is caused by bias within the sample. Considering that the fitness function was  $F(u, v, w) = u \cdot 9 + v \cdot 12 + w \cdot 30$ , we are getting a false positive for linkage between  $u$  and  $v$ .

The issue of false positives arising from biased distributions is particularly relevant to metaheuristics. **Hitch-hiking** causes “vestigial” traits to survive because they appear in solutions that already have beneficial traits. This phenomenon is discussed in more detail in Section 2.1.5

**PAIRED SAMPLES APPROACH** Here we measure the average *change* in fitness observed by modifying solutions.

- Within our samples, changing the value of  $u$  from 0 to 1 causes a +9 change in fitness on average.
- Solutions which have  $v = 0$  see an increase of +12 in fitness on average against their  $v = 1$  counterpart.
- The effect of switching  $(u, v)$  from  $(0, 0)$  to  $(1, 1)$  can only be seen in one solution, and it is +21.

This analysis suggests that  $u$  and  $v$  are not linked, since a +21 change in fitness is expected when combining a changes which cause +12 and +9 on their own.

This approach is much more **robust against unevenly distributed datasets**, since changes are measured *relatively*. For metaheuristic-derived data, paired samples methods are more reliable.

### 7.2.3 *Symmetric or Not Symmetric*

Linkage can be thought of as a symmetric relationship between pairs (or sets) of variables, where the variables depend on each other equally, but this is not always the case. For example, there may be a variable with a constant effect on fitness, and another variable that has an effect which depends on the former variable.

Methods which are non-symmetric are often represented as directed graphs, such as MIMIC [70], Bayesian Optimisation Methods [51, 216], and Bivariate Marginal Distribution Algorithm (BMDA) [219]. This is in contrast to methods which use undirected graphical models, such as Extended Compact Genetic Algorithm (EcGA) [118], Distribution Evaluation Using Markov network [241], and GOMEA [265].

### 7.2.4 *Whether it is probability of linkage, or strength of linkage*

There is a subtle difference in how linkage can be quantified. A scalar value might indicate the **confidence, or probability** that a given set of variables displays linkage effects. Other times, the scalar value might indicate how “**strongly**” linkage affects observations. A similar distinction is seen in statistical tests: the p-value and the effect size are not equivalent. If the p-value is very low, it means that there is a high chance of a difference in means, regardless of how large that difference is.

This difference is not to be overlooked, since “higher linkage” might mean “certainty” or it might mean “high weight on fitness”. In the case of graphical linkage models, probability measures are more common since they can be more easily collapsed into “linkage” and “non-linkage”. In machine learning, the researcher might instead view *all* variables being linked, and they aim to find the set which is *the most strongly* linked [115, 157].

### 7.2.5 *Differing definitions of non-linkage*

Different linkage detection methods use different definitions of non-linkage. It is important to discuss this, in order to show that non-linkage under a certain definition often does not apply universally.

The most common assumption in the literature is that independent variables are those which are **linearly independent**, i.e. their combined effect is simply the sum of their individual effects. This assumption only holds for additively separable functions, and thus it is not generally applicable [112].

Other times, the methodology is based on the **probability** of events. If two events have probabilities  $p_a$  and  $p_b$  respectively, the probability of them happening together is  $p_a \cdot p_b$  when they are independent.

Instead of probability, some methods think in terms of **information content**, where each knowing the value of a variable provides a measurable quantity of **information**. If two variables provide information  $H_a$  and  $H_b$  individually, they are expected to provide  $H_a + H_b$  when both are known and these are independent [136].

## 7.3 OVERVIEW OF LINKAGE LEARNING METHODS

A brief state of the art of linkage learning methods is provided, grouped by 3 categories: **non-linearity detection methods**, **fitness-independent methods** and **novel methods**.

Each method is discussed using the labels defined in this chapter, along with motivations, how non-linkage is defined, the main advantages and disadvantages. Table 7.4 summarises their key properties and provides information which should be considered to determine when each method is suitable.

### 7.3.1 Non-linearity Detection Methods

These methods detect linkage by checking whether the fitness function is non-linear with respect to pairs of variables. For these problems, non-linkage is when variables each contribute **linearly** to the fitness function.

This family of methods shares a fundamental limitation: they conflate non-linearity with variable interaction, which causes both false positives and false negatives (as shown in Section C.3), especially as they rely on samples rather than complete datasets.

This raises the question of whether these methods provide a sufficiently general framework. An in-depth discussion of this issue is beyond the scope of this chapter, so it is provided in Section C.3. Despite these concerns, these methods are the most commonly implemented thanks to their simplicity and similarity to existing concepts.

- **Perturbation-based linkage** [202, 127] (defined in Section C.4). This method is the most well-known, since it resembles how a human would calculate linkage by hand. Given a solution, the method will apply small changes to the solution (perturbations) and observe how its fitness changes. Two variables are independent when each adds to fitness independently, i.e. the combined perturbation equals the sum of the individual ones. **Linkage strength is how much the combined perturbation's effect departs from the sum of the effects of individual perturbations.** This is the only paired-samples approach that sees common use in the literature. Its main limitation is that it **requires live fitness evaluations**: it cannot work from a static dataset, and the number of evaluations grows quadratically with the number of variables.
- **Walsh decomposition** [127, 107, 128]. Perturbation is defined for pairs of variables, but it is generalised into higher order linkage by Walsh decomposition [127]. Every binary fitness function can be expressed as a Walsh function: a sum of terms, each relating to a combination of values of a subset of variables. **The coefficient of a term is the linkage strength for that subset**, so two variables are independent when their coefficient is zero. Walsh decomposition has many advantages over perturbation: it works from a static dataset, requires no new fitness evaluations, and captures interactions of any order, (although it requires paired samples to be present already in the sample of solutions). Therefore, it is technically an independent samples method. Its key limitation is that **it only works on binary problems**.
- **Linear models with interaction terms** [8, 160] (Section C.3). The aforementioned methods can only work with combinatorial variables, while **linear models with interactions** can be applied to continuous variables as well.

A linear model of the form  $\text{Fitness} \sim a + b + a:b$  is fitted to the dataset, where  $a, b$  are solution variables. **The coefficient of the interaction term  $a:b$  is the linkage strength**, and two variables are independent when this coefficient is zero. This method and all derivations are independent samples methods. Linear model building assumes that the dataset is uniform or normally distributed, so biased datasets are likely to skew the resulting coefficients.
- **ANOVA** [8, 160, 144, 90, 131] (Section C.5). ANOVA fits the same linear model as above, but instead of reporting the coefficient of the interaction term it reports **the probability that the coefficient is non-zero** (a p-value), along with an effect size ( $\eta^2$ ). Therefore, this method can both report the probability of linkage (even accounting for the quantity of data available), and its strength. Type II ANOVA is recommended, as it is the most robust for unbalanced datasets [169]. This method inherits the limitations of linear models.
- **SHAP / XGBoost linkage** [186, 50] (Section C.6.1). Variable importance can be generalised to calculate the importance of *pairs* of variables. By training an XGBoost model to predict solution fitness, **SHAP interaction values can be extracted** to quantify how much each pair of variables contributes. (XGBoost models lend themselves well to the specific calculation required). These are averaged over all solutions to obtain a global linkage estimate. SHAP interaction is theoretically a non-linearity detection method, but the presence of the model amortises the issues associated with those methods. This method proves to be highly accurate at determining linkage (which will be seen in the evaluation), but this comes at the expense of time - training the model becomes unfeasibly slow

for large quantities of solutions and high-dimensional problems (shown in the evaluation in Section 7.5.6.1).

- **Friedman’s H statistic** [95] (Section C.10). This method is born from XAI theory relating to Partial Dependence plots, where the fitness function is modelled as  $F(x) = \hat{F}_j(x_j) + \hat{F}_k(x_k) + \hat{F}_{jk}(x_j, x_k)$ , usually by using a ML model. For pairwise combinatorial linkage, **no surrogate model is needed**:  $\hat{F}_j$  and  $\hat{F}_k$  are derived from averages, and the resulting approximation is similar in structure to the linear model previously discussed, but it is inferred differently. The H statistic is the variance of the interaction term  $F_{jk}$ , normalised by the variance of the full two-variable model. Two variables are not linked when (in a model which uses just them for predicting fitness) all the variance is explained by their univariate variance. It is faster and more direct than SHAP, but it remains a non-linearity detection method based on independent samples.

### 7.3.2 Fitness independent methods

These methods study the “conditional distribution” of the variable’s values, and ignore fitness values. Therefore, the results depend entirely on the biases in the dataset determined by how the data was collected (e.g. the metaheuristic, convergence level or hitch-hiking effects). The main advantage for these is that they do not require the fitness values, which makes them faster to compute and more widely applicable. This comes at the cost of reduced linkage predictive power, and cannot work on *non-biased* datasets. Both methods shown here are independent samples approaches.

- **Mutual Information and Bayesian methods** [218, 216, 136] (Section C.9). Mutual Information (MI) quantifies how much knowing one variable’s value gives information about another’s, using the joint and marginal distributions estimated from the dataset. Bayesian methods [218, 216] build on information theory to construct directed graphical models of variable dependencies.
- $\chi^2$  **tests** [238] (Section C.8). This method provides a statistical test for combinatorial datasets, so that it can provide both a p-value and an effect size (Cramer’s V). It compares the joint probability distribution of a pair of variables against what would be expected if they were independent. Two variables are unlinked when their joint distribution is consistent with independence.

### 7.3.3 Other Methods

There are many domain-specific ways of implementing linkage detection. One which was used within this thesis was **cosine similarity** for PS4XRS (discussed in Chapter 8 and published in [43]).

There, a model associates each variable of the solution with an embedding vector, and cosine similarity measures how “similar” two variables are. Two variables are unlinked when their embedding vectors are orthogonal.

## 7.4 NOVEL LINKAGE METHODS

Having found existing linkage methods to be unsuitable (which will be shown in Section 7.5), the implementation of the atomicity objective in this thesis required the creation of linkage learning methods that are both fast and accurate. This led to the creation of two linkage learning methods: **Mutual Information of Survival** and **Dominance-based linkage**.

### 7.4.1 Mutual Information based on survival probability

In fitness-independent methods, we analyse the solutions that “survived” for long enough that they were added into a population of solutions. These methods are therefore very flexible (they make few assumptions), but the dependency on just the distribution of solutions has its faults. Instead, we propose to reintroduce the fitness values into the procedure to make them applicable to any dataset of solutions, such as a uniformly randomly generated sample.

This section will propose a method that is summarised as follows:

1. We use the solutions contained in the  $P_{\text{Ref}}$ .
2. For each “gene” we can **estimate the probability** of it being selected by a tournament operator. This is implemented by the function `survival( $x^\psi$ )` and is based on fitness values.
3. Using `survival`, we produce estimated conditional distributions, and use Mutual Information to quantify linkage strength.

Therefore, this method belongs in the independent samples, fitness-informed, symmetric, strength-of-linkage methods.

**THE SURVIVAL FUNCTION** We begin by using a dataset of solutions alongside their fitness values (the  $P_{\text{Ref}}$ ). We pretend to select a single individual from this population by using a theoretical fitness-proportionate selection. This is used to derive a formula that gives a “survival” probability to each solution, shown in Equation (7.2). This is obtained by the scaling steps required to implement fitness-proportionate selection (first force all the fitness values to be in the range  $[0, 1]$  in Equation (7.1), then divide them by their sum so that the sum of the probabilities is 1).

$$F\_range\_01(x) = \frac{F(x) - \min(F(P_{\text{Ref}}))}{\max(F(P_{\text{Ref}})) - \min(F(P_{\text{Ref}}))} \quad (7.1)$$

$$\text{survival}(x) = \frac{F\_range\_01(x)}{\text{sum}(Fs\_range\_01(P_{\text{Ref}}))} \quad (7.2)$$

The `survival` function can then be extended to work on patterns as well: `survival( $x^\psi$ )` returns the probability of the selected solution containing  $x^\psi$  (Equation (7.3)).

$$\text{survival}(x^\psi) = \text{sum}(\text{survival}(P_{\text{match}})) \quad (7.3)$$

`survival(( * * 1 * ))` will produce the probability of the selected solution having the third variable equal to 1. This can be used to find joint probabilities: `survival(( * * 1 1 ))` finds the probability of the last two variables being equal to 1.

#### 7.4.1.1 Mutual Information Survival linkage

The `survival` function provides all the elements required for Mutual Information, which is then put together to define MISurvival (Equation (7.5), similarly to how it is defined in DSMGA-II [136] but now using fitness values through `survival`.

$$\text{MISurvival}(a, b) = \sum_{\substack{\text{val}_a \in \text{options}(a) \\ \text{val}_b \in \text{options}(b)}} \text{MI}(p_a, p_b, p_{ab}) \quad (7.4)$$

where  $e = \text{****} \dots$

$$\begin{aligned} p_a &= \text{survival}(e_{a=\text{val}_a}) \\ p_b &= \text{survival}(e_{b=\text{val}_b}) \\ p_{ab} &= \text{survival}(e_{\substack{a=\text{val}_a \\ b=\text{val}_b}}) \\ \text{MI}(x, y, z) &= z \cdot \log\left(\frac{z}{xy}\right) \end{aligned} \quad (7.5)$$

(Note that in Equation (7.5) we use the notation  $e_{i=j}$  to mean a *PS* where all the variables are *(\*)* except for variable  $i$  being set to the value  $j$ ).

In summary, this method implements a linkage method based on information theory, but it is more robust against biased distributions because it uses fitness values.

#### 7.4.2 Dominance based Linkage

This method is a paired-samples approach which aims to be more efficient and generalised than Perturbation. The word “**dominance**” used here is borrowed from multi-objective optimisation, where it is used to mean that a solution is *better* than another one. Dominance-based linkage analyses which solutions are **better** than others (hence “dominance”), and does not use the magnitude of the difference between fitness values.

This makes less assumptions about the nature of the linkage relationship (e.g. that non-linearity is treated as linkage), which makes it more general than other methods.

Consider an example fitness function  $F_{\text{link}}$  (defined in Equation (C.1) but it is not necessary to know it now). Then, consider the following pairs of solutions, which differ only by the first variable. We compare their fitness when  $a = 0$  and when  $a = 1$ :

|                                |    |                                 |                       |
|--------------------------------|----|---------------------------------|-----------------------|
| $F_{\text{link}}(0, 0, 0) = 0$ | vs | $F_{\text{link}}(1, 0, 0) = 10$ | $\Rightarrow$ greater |
| $F_{\text{link}}(0, 0, 1) = 0$ | vs | $F_{\text{link}}(1, 0, 1) = 15$ | $\Rightarrow$ greater |
| $F_{\text{link}}(0, 1, 0) = 2$ | vs | $F_{\text{link}}(1, 1, 0) = 8$  | $\Rightarrow$ greater |
| $F_{\text{link}}(0, 1, 1) = 3$ | vs | $F_{\text{link}}(1, 1, 1) = 12$ | $\Rightarrow$ greater |

In all situations, the solution was improved in fitness by changing  $a$  from 0 to 1, therefore we can say that the effect of this variable *depends* on no other variables (in terms of sign). Now consider the following paired samples, where we compare when  $b = 0$  against when  $b = 1$ :

|                                 |    |                                 |                       |
|---------------------------------|----|---------------------------------|-----------------------|
| $F_{\text{link}}(0, 0, 0) = 0$  | vs | $F_{\text{link}}(0, 1, 0) = 2$  | $\Rightarrow$ greater |
| $F_{\text{link}}(1, 0, 0) = 10$ | vs | $F_{\text{link}}(1, 1, 0) = 8$  | $\Rightarrow$ lower   |
| $F_{\text{link}}(0, 0, 1) = 0$  | vs | $F_{\text{link}}(0, 1, 1) = 3$  | $\Rightarrow$ greater |
| $F_{\text{link}}(1, 0, 1) = 15$ | vs | $F_{\text{link}}(1, 1, 1) = 12$ | $\Rightarrow$ lower   |

These pairs show that sometimes changing  $b$  from 0 to 1 is beneficial to fitness, and sometimes it is counterproductive. With further analysis, it can be seen that when  $a = 0$ , then  $b = 1$  is always better, whereas when  $a = 1$  the best value is attained with  $b = 0$ .

This methodology can detect **asymmetric** linkage, giving more granularity: note how  $b$  depended on  $a$ , but not vice-versa.

Let us now define this linkage, first by giving an informal breakdown:

For a variable  $a$  we can collect paired samples of solutions which differ only by that variable. Each change of  $a$ , from a value  $u$  to a value  $v$ , will result in a new fitness, which may be **better**, **worse** or **tied**. We call this outcome “**dominance**”. The frequency of these outcomes can be counted, for example “changing  $u$  to  $v$  results in **better** 20 times, **worse** 10 times, and **tie** 2 times”.

A variable  $b$  can be used to partition these frequencies. For example, “when  $b = 1$ ,  $u \rightarrow v$  has outcomes **better**: 8 times, **worse**: 0 times, and **tie**: 1 time.”.

If the variable  $a$  is not linked in any way to  $b$ , then **these partitionings should resemble the unpartitioned distribution**. If they are linked, then the distribution changes significantly based on the value of  $b$ .

We check what is the probability of these smaller distributions originating from the unpartitioned distribution, using a  $\chi^2$  test.

The advantages of this approach are:

1. It is NOT a non-linearity detection method, and uses change-of-sign effects. This allows it to make fewer assumptions about the fitness function and avoid the issues discussed in Section C.3.
2. It does NOT use any additional function evaluations.
3. Applicable in scenarios where solutions are not a total-order, such as a Multi-Objective Optimisation problem, because “dominance” can be generalised to be “Pareto dominance”.

Finding the set of pairs of solutions that differ by one variable can be computationally slow, but by implementing an algorithm that exploits hash-value collisions it can be made efficient enough that its execution time is negligible. The algorithm is presented in Chapter B.

Because dominance-based linkage only requires solutions to have a “dominance” relationship, it could be easily applied to multi-objective contexts by using Pareto-dominance. This has not been pursued in this chapter, but it is a possible avenue for future research.

#### 7.4.2.1 Definition of Dominance linkage

This section will define Dominance-based linkage. From the set of solutions in  $P_{\text{Ref}}$ , we find a list of all the pairs of solutions differing by one variable (implemented by `paired_solutions`, defined in Chapter B). The output of this function is a set of pairs of solutions. We then define the following function:

$$\begin{aligned} \text{greater\_count}(a, \text{val}_1, \text{val}_2, b, \text{val}_b) = & |\{F(x) > F(w) \\ & s.t. (x, w) \in \text{paired\_solutions}(P_{\text{ref}}), \\ & x_a = w_a, \\ & x_a = \text{val}_1, w_a = \text{val}_2, \\ & x_b = w_b = \text{val}_b\}| \end{aligned} \quad (7.6)$$

This will count how many times, in the paired samples present in the  $P_{\text{Ref}}$  which differ only by  $a$  and where  $b = \text{val}_b$ , the solution with  $a = \text{val}_1$  has greater fitness than the solution with  $a = \text{val}_2$ .

Similarly, we can define `lower_count` and `tie_count`:

$$\text{lower\_count}(a, \text{val}_1, \text{val}_2, b, \text{val}_b) = |\{F(x) < F(w) \text{ s.t. } \dots\}| \quad (7.7)$$

$$\text{tie\_count}(a, \text{val}_1, \text{val}_2, b, \text{val}_b) = |\{F(x) = F(w) \text{ s.t. } \dots\}| \quad (7.8)$$

These functions are then used to make a table representing the aforementioned “partitions” for a variable  $a$  being tested for dependence on  $b$ , where:

1. Each row header is a possible value of  $b$
2. Each column is a triplet  $(\text{val}_1, \text{val}_2, \text{dominance})$ .  $\text{val}_a$  and  $\text{val}_b$  are possible values of  $a$  ( $\text{val}_1 < \text{val}_2$  to ignore duplicates), and  $\text{dominance} \in \{\text{greater}, \text{lower}, \text{tie}\}$ .

This table will be referred to as the `Dominance_partition_table(a, b)`. An example of what such a table might look like for RoyalRoad is shown in Table 7.2, where the two variables being analysed are independent. To note is that the proportions for each row remain roughly constant  $(\frac{1}{8}, \frac{7}{8}, 0)$  regardless of which  $\text{val}_b$  is used.

Compare this against a table for two variables which are linked, in Table 7.3. Here, the distribution changes, which is detected by a  $\chi^2$  test producing a p-value of  $3 \cdot 10^{-7}$  and Cramer’s V of 0.384 (which is relatively high).

Table 7.2: Outcome partition table where  $a$  and  $b$  are independent, in a RoyalRoad-Problem

| $(\text{val}_1, \text{val}_2, \text{Dominance}) \rightarrow$ | $(0, 1, \text{greater})$ | $(0, 1, \text{tie})$ | $(0, 1, \text{lower})$ |
|--------------------------------------------------------------|--------------------------|----------------------|------------------------|
| $(\text{val}_b) \downarrow$                                  |                          |                      |                        |
| 0                                                            | 15                       | 105                  | 0                      |
| 1                                                            | 10                       | 70                   | 0                      |

Finally, we can define two metrics using this kind of linkage:

$$\text{paired\_dominance\_pvalue}(a, b) = \quad (7.9)$$

$$\chi^2\text{Test}(\text{Dominance\_partition\_table}(a, b)).\text{pvalue}$$

$$\text{paired\_dominance\_strength}(a, b) = \quad (7.10)$$

$$\chi^2\text{Test}(\text{Dominance\_partition\_table}(a, b)).\text{CramersV}$$

$$(7.11)$$

Table 7.3: Outcome partition table where  $a$  and  $b$  are linked, in a RoyalRoad problem

| $(\text{val}_1, \text{val}_2, \text{outcome}) \rightarrow$ | $(0, 1, \text{greater})$ | $(0, 1, \text{tie})$ | $(0, 1, \text{lower})$ |
|------------------------------------------------------------|--------------------------|----------------------|------------------------|
| $(\text{val}_b) \downarrow$                                |                          |                      |                        |
| 0                                                          | 0                        | 100                  | 0                      |
| 1                                                          | 23                       | 69                   | 0                      |

A  $\chi^2$  test is used because it can determine the likelihood of discrete distributions being samples from the same overall distribution. From this, we can obtain a p-value to determine the probability of pairwise linkage, and Cramer's V to quantify its strength.

Cramer's V was chosen as the measure of strength because it is the only metric of its kind to be applicable to tables bigger than  $2 \times 2$  (i.e., it works for variables which are not binary) [239], and its interpretation is discussed in [161]. In general, it is a conservative measure, and values above 0.10 can be considered significant.

**TIME COMPLEXITY** The algorithm which finds all the paired samples was designed to have fast time complexity:  $\mathcal{O}(NL)$  (where  $N$  is the quantity of solutions in the  $P_{\text{Ref}}$ ). To calculate linkage, the algorithm simply goes through all the pairs found (the quantity of which depends on the  $P_{\text{Ref}}$  but it is almost always below  $NL$ ) and tallies up the dominance scenarios. Therefore, the final algorithm is dominated by the  $\mathcal{O}(NL)$  preprocessing step.

Table 7.4: Overview of linkage learning methods.

| Kind                          | Fitness-informed? | Paired samples method? | Non-linearity method? | Non-linkage                                                                                                           | Pros                                                                  | Cons                                                                                                      |
|-------------------------------|-------------------|------------------------|-----------------------|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| Perturbation                  | ✓                 | ✓                      | ✓                     | Combined perturbation = sum of individual perturbations                                                               | Easy to understand<br>Solution specific                               | Requires additional fitness function evaluations<br>Speed scales poorly                                   |
| Walsh                         | ✓                 |                        |                       | Walsh coefficient is zero                                                                                             | Can capture higher-order interactions                                 | Only works on binary problems                                                                             |
| Linear models                 | ✓                 |                        | ✓                     | Interaction term coefficient is zero                                                                                  | Generalises Walsh.                                                    | One-hot encoding is sometimes required and leads to large unstable models<br>Biased datasets skew results |
| ANOVA                         | ✓                 |                        | ✓                     | p-value of interaction term is not significant                                                                        | Can detect when there is not enough data                              | All limitations of linear models apply                                                                    |
| SHAP XGBoost                  | ✓                 |                        |                       | SHAP interaction value for pair is 0                                                                                  | Uses a Surrogate model, which can capture non-linear effects.         | Requires the training of a model, which makes it slow.<br>Linkage reflects the model                      |
| Friedman H                    | ✓                 |                        |                       | Considering a variable pair adds no variance ( $H \sim 0$ )                                                           | Faster and more direct than SHAP, no model required                   |                                                                                                           |
| Chi-squared test              |                   |                        |                       | Joint distribution matches product of marginals                                                                       | Produces p-value and effect size                                      | Results are based on sampling bias                                                                        |
| Mutual Information / Bayesian |                   |                        |                       | $MI(a, b) = 0$ , joint distribution factors into marginals                                                            | Can build Bayesian Graphical models                                   | Results are based on sampling bias                                                                        |
| MI on survival probability    | ✓                 |                        |                       | Considering solutions with high fitness, knowing one variable's value does not give more information about the other. | Uses fitness to reduce dependence on sampling bias                    | Not entirely immune from dataset bias                                                                     |
| Dominance-based linkage       | ✓                 | ✓                      |                       | The Dominance distribution of one variable is not affected by the value of the other variable.                        | False positives are harder to obtain<br>Can detect asymmetric linkage | Requires enough solutions pairs differing by one variable.                                                |

There are few experimental studies measuring the effectiveness of different linkage discovery techniques. Most research develops formulas to detect linkage, applies it within EDAs and then report the performance of the metaheuristic. Therefore, linkage is almost always a tool for improved search performance and does not get evaluated on its own, as seen in [117, 59, 241, 264, 266, 136, 215, 70, 218, 38] for example.

Proof that linkage is being learned accurately is from anecdotal evidence (such as [202, 127, 296, 269]) rather than via replicable and objective evaluation methods. Thus it is not known how different methods compare to each other, and if they perform best on certain classes of problems.

This section will aim to provide a methodology to compare different linkage learning methods, in the context of metaheuristics. Knowing which linkage methods produce the best results is necessary to implement an **atomicity** metric that accurately scores **PSs**.

The following research questions are investigated here:

**LRQ1 Which linkage method performs best in general, when treated as a classification prediction task evaluated via balanced accuracy?**

**LRQ2 Does using data from different metaheuristics affect linkage learning?**

**LRQ3 Which problems are hard for linkage learning?**

**LRQ4 Which linkage methods are susceptible to non-linearity?**

**LRQ5 Which linkage methods have the best  $\mathcal{O}$  time complexity?**

#### 7.5.1 Challenges in evaluation, and their resolution

Linkage learning has been implemented in many ways, which are not always directly comparable. Here, we will look at them under the same scenario: using data from a metaheuristic, we obtain scalars quantifying the linkage between pairs of variables.

**DIFFERENT INFORMATION USED** Fitness-independent methods ignore the fitness function, so it is expected that fitness-informed methods will have an advantage because they use more information, and Perturbation even requires new fitness function evaluations, so it has a further advantage. Some methods expect the sample of solutions to be uniform, while others require it to be biased. These differences make comparisons unfair, but it is unclear how these differences affect their performance.

Our experiments will consider all the categories of linkage learning discussed, and any known disadvantages will be acknowledged. This is to show empirically the differences in performance which are expected. The experiments which will be shown will use the  $P_{\text{Ref}}$  obtained from running different metaheuristics, including Random Search so that both uniform and “biased” datasets are considered.

**NO GROUND TRUTH FOR REGRESSION** Linkage detection is often thought of as a “regression” problem, where the task is to find the “correct” scalars. The issue lies in what is “correct”, which is highly dependent on the kind of fitness function being analysed. This is arguably a fallacy: the class of all possible variable interactions cannot be objectively mapped into scalars, and therefore there is no way of obtaining ground truth values for evaluation.

This evaluation will circumvent this issue by reframing the task as a **classification** problem. This choice is justified in the coming paragraph.

**LINKAGE AS A CLASSIFICATION TASK** There is a misalignment between how linkage is implemented and how it is conceptualised.

For a pair of variables, it is much easier to describe them as “linked” and “not linked” – a binary label rather than a scalar. Linkage is often represented as a graph, where nodes are either connected or not.

In implementation, the task is instead treated as a **regression** task: there is a scalar to be calculated. Implicitly, it is assumed that those scalars can be used for binary prediction by using a fixed threshold; one might say for example that two variables are linked if the p-value for their  $\chi^2$  test is below 0.05. These thresholds are often hypothetical and not tested or reported, but they are a crucial element in the overall system.

Our evaluation resolves the classification-regression dilemma by automatically finding thresholds, those which maximise **balanced accuracy**. The quality of a linkage learning method is its balanced accuracy in predicting if a given pair of variables is linked or not.

### 7.5.2 Evaluating linkage detection: automatic threshold selection

The proposed evaluation method is applicable to any linkage detection method that produces a scalar which measures linkage between pairs of variables. **There is no need to manually provide a threshold.**

**GROUND TRUTH AND PREDICTIONS** Firstly, a benchmark combinatorial optimisation problem is required, where for every pair of variables there is a “ground truth” class of linkage (linked or not linked). This ground truth comes by construction: the problems were all designed to display the intended linkage effect, which will be explained more in Section 7.5.3.

The problem is solved by a metaheuristic, which produces the data to be analysed. A linkage learning method is applied to this data, resulting in “predicted linkage”. The ground truth linkages and the predicted values are put into a table such as Table 7.5.

| Var A | Var B | (ground truth) linkage class | predicted linkage |
|-------|-------|------------------------------|-------------------|
| 0     | 1     | True                         | 0.2               |
| 0     | 2     | False                        | -0.4              |
| 1     | 2     | True                         | 0.1               |
| ...   | ...   |                              |                   |

Table 7.5: Example of a table to be used for evaluation of linkage. Every row corresponds to a pair of variables, for which we know if they are linked or not (True = linked, False = non-linked), along with the linkage strength that was measured.

**AUTOMATIC THRESHOLD SELECTION** Linkage scores can be converted into binary values (linkage vs non-linkage) by using a threshold, and then these binary values can be compared against the ground truth by using precision, accuracy and other quality metrics. Here, instead of manually picking the thresholds, **we pick the one which maximises balanced accuracy**. Balanced accuracy is a metric defined as the average of the accuracy on positive instances and the accuracy on negative instances.

This performance metric is chosen because:

- The quality metric should **not focus on positives or negatives specifically**, as a positive instance (linkage) is as important as a negative instance (independence). This excludes precision and recall, among others.
- There are many cases where **true instances (where there is linkage) are the minority**. If accuracy was used, the score will not reflect the performance on true instances fairly. Balanced accuracy exists to resolve this issue.

Moreover, automatically finding the threshold based on balanced accuracy has the following advantages:

- It removes the bias of manually picking a threshold.
- It does not disadvantage any method unfairly, since its best possible balanced accuracy is presented. Although some methods might underperform when using this metric, for this analysis we decided to focus on one metric to allow for general comparisons.
- It allows to detect the **stability** of a linkage detection method, which is discussed later.

More formally, we define the BBA (best balanced accuracy) of linkage scores as:

$$\text{BBA}(\text{linkage}, \text{ground}) = \max \left( \max_t \text{BA}(\mathbf{v}_{<t}, \text{ground}), \max_t \text{BA}(\mathbf{v}_{>t}, \text{ground}) \right) \quad (7.12)$$

where:

$$\begin{aligned} \mathbf{v}_{<t}, \mathbf{v}_{>t} &= \text{boolean vectors for linkage } < t \text{ and linkage } > t, \\ \text{ground} &= \text{ground truth boolean linkage} \\ \text{BA}(\hat{\mathbf{y}}, \mathbf{y}) &= \text{balanced accuracy of predictions } \hat{\mathbf{y}} \text{ against } \mathbf{y}. \end{aligned} \quad (7.13)$$

Note the presence of  $\mathbf{v}_{<t}$  and  $\mathbf{v}_{>t}$ : this is to account for scenarios where the linkage score is *above* a threshold and when it is *below* a threshold. Finding the values of  $t$  which maximise the balanced accuracy is done through an exhaustive search.

**MEASURING STABILITY** Imagine that we solve an optimisation problem using a GA and use the resulting  $P_{\text{Ref}}$  as the data to predict linkage. This data is going to be different between separate GA runs, and we might want to know if our linkage method produces “stable” results. It is hard to define what stable would mean, but here we use a simple idea: a linkage learning method is **stable if there exists a fixed threshold that makes it perform well (in terms of best balanced accuracy) across different sources of data.**

To measure this empirically in our GA example, we could simply produce tables similar to Table 7.5 for each GA run, concatenate them vertically and then apply the evaluation described above, reporting the best balanced accuracy.

If the resulting best balanced accuracy is low, it means that no fixed threshold worked well across the different datasets, and thus the linkage learning method is unstable. It cannot be reliably used to distinguish linkage and non-linkage.

Similarly, we can compute the BBA over different problems and metaheuristics in order to evaluate a method’s general applicability, as shown in Figure 7.2.

### 7.5.3 Benchmark problems used

The evaluation presented will be applied to 4 benchmark problems, which aim to represent a diverse set of linkage kinds:

- **RoyalRoad** (3 cliques, 5 bits each) (defined in Section 2.2.2.1): Linkage is observed only when all the values of a clique have certain values. Therefore, linkage is a “rare” phenomenon
- **Trap-k** (3 cliques, 5 bits each) (defined in Section 2.2.2.2): Linkage is similar to RoyalRoad in how it is rare, but it is complicated by the presence of non-linked behaviour (each zero added contributes linearly to fitness in most cases).

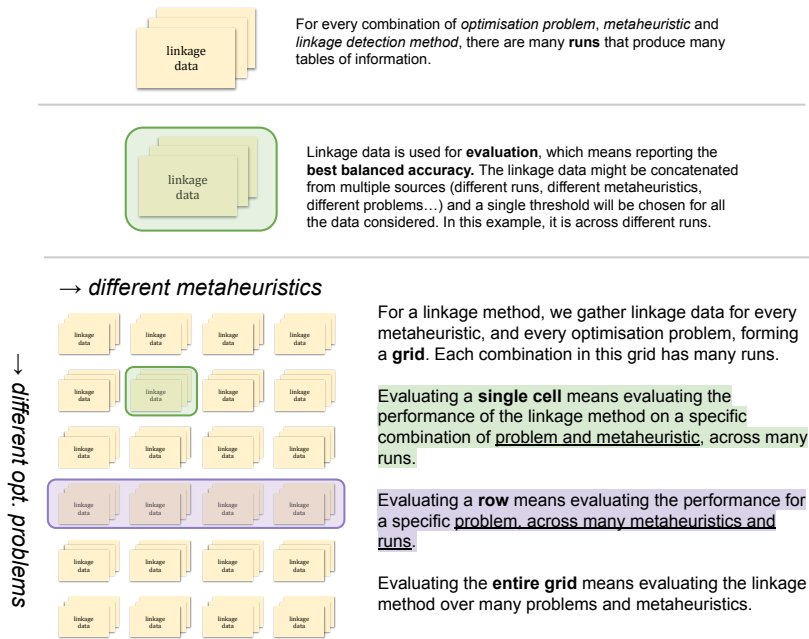


Figure 7.2: Evaluation strategy for linkage detection. Each yellow rectangle is a table with ground truth linkage information and measured linkage values.

- **Parity Problem** (3 cliques, 5 bits each) (defined in Section 2.2.2.3): Linkage is always affecting the fitness, but through a higher-order relationship which is hard to capture. This problem is known to be hard to model through bivariate measurements [60].
- **Graph Colouring** (10 variables, 3 colours, graph shown in Figure 7.3) (defined in Section 2.2.3): linkage is often represented as a connected graph, so using Graph Colouring is a natural choice. This problem allows the creation of different structures, such as cliques of different sizes, and partially connected cliques (as shown in Figure 7.3).

Non-linearity detection methods are expected to perform poorly on problems which are known to be non-linear. To test this, a “Sigmoid” version of each of the above problems is also included - the original fitness function  $F(x)$  is replaced by  $\sigma(F(x))$ , where  $\sigma$  is the non-linear Sigmoid function (discussed in Section C.3). Any monotonic

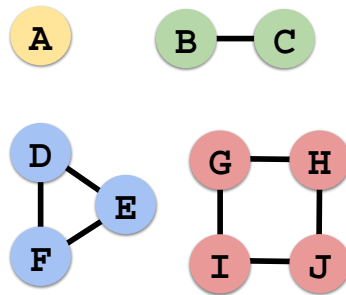


Figure 7.3: Graph Colouring problem instance used for evaluation. The problem contains variables that are in differently sized cliques, and even variables that are not connected to anything.

increasing non-linear function would have fit this purpose (since the ordering of solutions would not have been affected), but  $\sigma$  was chosen due to its well-known role in ML of converting linear functions into non-linear functions.

As discussed in Section 7.5.1, these problems are chosen because ground truth can be easily inferred from their definitions. The ground truth for each problem is shown in Figure 7.4, with their sigmoid counterparts being identical.

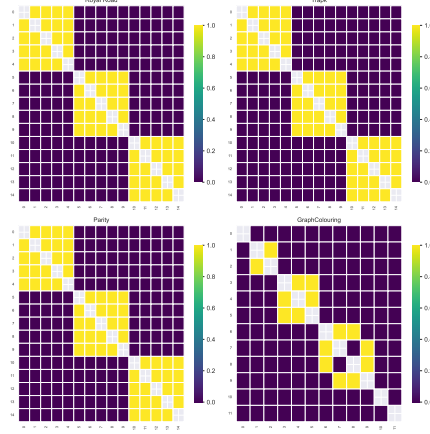


Figure 7.4: Ground truth linkage for benchmark problems

#### 7.5.4 Metaheuristics used

The following 4 metaheuristics were used: a traditional GA (to represent population based methods), SAs (to represent single-individual methods), Tabu search (to represent diversity preserving methods), and random search (to remove any bias caused by the metaheuristic).

Each combination of metaheuristic, optimisation problem and linkage detection method is executed 30 times with different seeds. The parameters used are as follows:

- **GA**: uniform mutation operator, mutation rate of  $\frac{1}{L}$ , crossover probability = 0.5, simulated binary crossover, population size = 100;
- **SA**: uniform mutation operator, mutation rate of  $\frac{1}{L}$ ;
- **Tabu search**: uniform mutation operator, mutation rate of  $\frac{1}{L}$ , Tabu list size = 30;
- **Random search**: no parameters.

Each of these algorithms was executed with a fitness evaluation budget of 10000, so that the  $P_{\text{Ref}}$  used is of size 10000.

In future work, the metaheuristics to be used for the evaluation should vary in what operators they use, whether crossover is present, whether it is population-based or not, whether it is an Estimation of Distribution Algorithm (EDA), and many other aspects. The evaluation presented here is preliminary.

The purpose of the metaheuristic is to generate the  $P_{\text{Ref}}$  from which linkage is learned. This is then the same  $P_{\text{Ref}}$  used for PS mining.

### 7.5.5 Linkage methods being evaluated

The linkage methods to be compared are:

- **Perturbation** (since it uses the live fitness function, it has an advantage over other methods). Follows definition in Section C.4;
- **ANOVA** (p-value and the effect size are treated as different metrics). Follows definition in Section C.5, where a linear model with an interaction term is fitted via ordinary least squares, and then a type II ANOVA test is carried out;
- **H-statistic**, as defined in Section C.10;
- $\chi^2$  **test** (p-value and effect size  $\eta^2$  are treated as different metrics) as defined in Section C.8;
- **Mutual Information** as defined in Section C.9;
- **Mutual Information of survival** as defined in Section 7.4.1;
- **Dominance linkage** (p-value and effect size  $\eta^2$  are treated as different metrics) as defined in Section 7.4.2;
- **SHAP**, based on an XGBoost model as defined in Section C.6.1 (100 estimators, with maximum depth of 6,  $\eta = 0.3$ ,  $\gamma = 0$  as suggested on documentation for the model [293] );

This list excludes Walsh and Linear models, as these are equivalent (for pairs of variables) to Perturbation and ANOVA respectively. Potential advantages and disadvantages due to fitness-function use are summarised in Table 7.4.

### 7.5.6 Experimental results

The predictive performance of each linkage method is compared here, split by benchmark problems and metaheuristics used.

The tables following do not show averages, but rather they are the overall predictive performance on given datasets. For example, Perturbation for RoyalRoad having a BBA (best balanced accuracy) of 80% “over all metaheuristics” would mean that “when given data generated from any of the metaheuristics considered, on any of their repeated runs, and about any two variables, we have 80% balanced accuracy when predicting linkage or non-linkage using Perturbation”.

**IN GENERAL** It is easier to start by comparing the performance of linkage methods, regardless of the optimisation problem or metaheuristic. The results in Table 7.6 indicate that **Perturbation yields the best balanced accuracy** overall (84.9%). This is not a surprise, since it is the only metric which requires additional fitness function evaluations. The second best linkage detection method was **SHAP** (69.9%). The worst approaches were those based on fitness-independent methods, which match expectations.

**ON EACH BENCHMARK PROBLEM** The next set of results relates to the performance on each benchmark problem, which is presented in Table 7.7. The main insights from this table are:

- The performance of linkage detection is affected by the benchmark function. For some it is nearly perfect, whereas for others it is nearly as bad as a coin toss. **Sigmoid Trap-k**, **Parity** and **Sigmoid Parity** are the benchmarks where the lowest performance is observed.
- There is a difference in performance between problems and their **Sigmoid** counterpart, but this effect is not observed everywhere. Perturbation is much more affected than any other method, which might be due to its non-linearity detection. For almost all linkage methods Sigmoid Trap-k is more challenging than Trap-k.

| Linkage method           | BBA          |
|--------------------------|--------------|
| Perturbation             | <b>84.9%</b> |
| SHAP                     | 69.9%        |
| Dominance (Cramer's V)   | 69.7%        |
| ANOVA ( $\eta^2$ )       | 69.5%        |
| Dominance (p-value)      | 69.2%        |
| ANOVA (p-value)          | 67.6%        |
| MI of survival           | 64.6%        |
| H statistic              | 62.6%        |
| Chi Squared (Cramer's V) | 60.5%        |
| Mutual Information       | 59.1%        |
| Chi Squared (p-value)    | 57.9%        |

Table 7.6: The best balanced accuracy for each linkage detection, when using data from every benchmark problem and every metaheuristic. The entries are sorted from best to worst

- Overall, **SHAP and Perturbation have the highest performance**, since both achieve more than 80% BBA on all problems except for Sigmoid Trapk.
- Dominance using Cramer's V is rounded in its performance, with only Parity problems having a score below 77%. It is the only method whose performance on Trap-k is the same as on Sigmoid Trap-k. This confirms that its design aims have been satisfied, but it is in general worse than Perturbation and XGBoost SHAP.
- MI of Survival performs poorly on Sigmoid Trap-k, Parity and Trap-k parity. This might indicate that it is a non-linearity detection method, although its implementation would not suggest so. It does outperform traditional Mutual Information, but by 10% in 2 cases and less than 5% in 6 cases.

| Linkage Method           | Royal Road  | Sigmoid Royal Road | Trapk        | Sigmoid Trapk | Parity      | Sigmoid Parity | GraphColouring | Sigmoid GraphColouring |
|--------------------------|-------------|--------------------|--------------|---------------|-------------|----------------|----------------|------------------------|
| Perturbation             | <b>100%</b> | 87.4%              | <b>98.7%</b> | 61.2%         | <b>100%</b> | 87.5%          | <b>100%</b>    | 82.5%                  |
| SHAP                     | 99.9%       | <b>99.8%</b>       | 91.2%        | 53.4%         | 92.1%       | <b>91.2%</b>   | <b>100%</b>    | 85.6%                  |
| Dominance (Cramer's V)   | 87.2%       | 86.3%              | 78.9%        | <b>77.0%</b>  | 57.8%       | 57.8%          | 86.6%          | 86.8%                  |
| Dominance (p-value)      | 80.1%       | 79.2%              | 78.4%        | <b>77.0%</b>  | 57.8%       | 57.7%          | 86.0%          | 86.1%                  |
| ANOVA ( $\eta^2$ )       | 89.5%       | 88.4%              | 80.5%        | 54.2%         | 55.0%       | 53.8%          | 98.8%          | <b>99.1%</b>           |
| ANOVA (p-value)          | 83.4%       | 81.9%              | 82.2%        | 53.5%         | 55.0%       | 53.9%          | 98.3%          | 98.8%                  |
| MI of survival           | 76.8%       | 77.6%              | 64.2%        | 57.0%         | 55.5%       | 54.6%          | 81.6%          | 80.8%                  |
| H statistic              | 88.9%       | 88.0%              | 75.9%        | 55.8%         | 53.5%       | 53.2%          | 94.0%          | 93.2%                  |
| Chi Squared (p-value)    | 67.8%       | 66.9%              | 62.9%        | 58.8%         | 54.9%       | 54.1%          | 81.9%          | 82.5%                  |
| Chi Squared (Cramer's V) | 66.6%       | 67.4%              | 61.3%        | 57.1%         | 54.9%       | 54.2%          | 79.9%          | 80.5%                  |
| Mutual Information       | 66.6%       | 67.3%              | 61.2%        | 57.1%         | 54.9%       | 54.3%          | 80.1%          | 80.7%                  |

Table 7.7: The best balanced accuracy is presented for every combination of linkage detection method and every benchmark problem. In bold is the highest number of each column (i.e. for each problem).

ON EACH METAHEURISTIC Next, in Table 7.8 we evaluate each linkage method on each metaheuristic. For all metaheuristics tested, Perturbation is the best method. Few

methods perform best with an unbiased dataset (from Random Search): Perturbation and ANOVA (p-value); the other methods perform poorly on this data.

Fitness-independent methods perform poorly with Tabu Search and Random Search (since these lack the bias which makes them work), whereas most methods perform similarly well for the GA, SA and Tabu search.

| Linkage Method           | Genetic Algorithm | Simulated Annealing | Tabu Search  | Random Search |
|--------------------------|-------------------|---------------------|--------------|---------------|
| Perturbation             | <b>87.6%</b>      | <b>85.1%</b>        | <b>86.3%</b> | <b>89.7%</b>  |
| SHAP                     | 73.4%             | 72.5%               | 74.7%        | 67.1%         |
| Dominance (Cramer's V)   | 78.2%             | 75.3%               | 77.3%        | 54.1%         |
| Dominance (p-value)      | 76.5%             | 73.8%               | 81.0%        | 54.6%         |
| ANOVA ( $\eta^2$ )       | 68.4%             | 70.1%               | 71.9%        | 68.8%         |
| ANOVA (p-value)          | 68.1%             | 69.5%               | 72.0%        | 72.0%         |
| MI of survival           | 67.1%             | 70.7%               | 64.1%        | 57.1%         |
| H statistic              | 61.7%             | 63.1%               | 63.7%        | 63.4%         |
| Chi Squared (Cramer's V) | 67.0%             | 71.1%               | 56.0%        | 53.2%         |
| Chi Squared (p-value)    | 64.2%             | 66.5%               | 53.9%        | 50.6%         |
| Mutual Information       | 64.8%             | 68.7%               | 53.9%        | 54.3%         |

Table 7.8: Table showing the best balanced accuracy for each linkage detection method on each metaheuristic. Figures in bold are the highest for their column.

**THRESHOLDS** Finally, we provide a table of the thresholds which we found to be ideal for each linkage method, in Table 7.9. These thresholds produce the best balanced accuracies in Table 7.6.

Although these cannot be compared, the p-values are not consistent between  $\chi^2$ , ANOVA and Dominance methods. For  $\chi^2$  tests, the threshold is much lower than 0.05, which indicates that perhaps  $\chi^2$  tests are likely to cause false positives when using 0.05 as a p-value threshold. This was observed in [38], where  $\chi^2$  tests were found to easily cause false positives.

| Linkage method           | Threshold for linkage |
|--------------------------|-----------------------|
| Perturbation             | > 0.00624430          |
| Chi Squared (p-value)    | < 0.00000053          |
| Chi Squared (Cramer's V) | > 0.13238968          |
| ANOVA (p-value)          | < 0.01430075          |
| ANOVA ( $\eta^2$ )       | > 0.00658212          |
| Mutual Information       | > 0.00000001          |
| MI of survival           | > 0.01912785          |
| Dominance (p-value)      | < 0.06367036          |
| Dominance (Cramer's V)   | > 0.20822820          |
| SHAP                     | < 0.02876192          |
| H statistic              | > 0.00220281          |

Table 7.9: Best threshold (yielding the best BBA) for each linkage detection method. The “Threshold” column indicates the threshold that yields the best balanced accuracy. These thresholds are those which yield the best balanced accuracy shown in Table 7.6.

### 7.5.6.1 Run times

It is necessary to report the runtimes and scalability of each methodology discussed, in order to get a fuller idea of the accuracy-runtime trade-off.

The following tables were generated by creating RoyalRoad problems of increasing sizes (variable counts going from 10 to 200 in increments of 10), and solving via a GA with different evaluation budgets (5000, 10000, 15000 and 20000). Using the resulting  $P_{\text{Ref}}$ , the various linkage learning methods are applied and timed (with a timeout of 30 seconds due to time constraints). Each run is repeated 30 times. The timing included all the steps required by a method, with Dominance-based linkage requiring a pre-processing step where all paired samples are identified first.

The average times are shown in Figure 7.6, where darker values correspond to longer execution times. A graph showing the runtime with respect to variable count is presented in Figure 7.5, where the size of the data is kept fixed to 20000 solutions.

The main insights are:

- The runtime is affected by both the quantity of variables and the size of  $P_{\text{Ref}}$ . All methods become slower as those parameters increase, but the increase in runtime is more than linear with respect to variable count Figure 7.5.
- The methods with the worst time complexity are Perturbation, ANOVA and SHAP. This matches expectations: If  $L$  is the quantity of variables, and  $N$  is the quantity of solutions, perturbation is expected to have complexity  $\mathcal{O}(LN^2)$ , while SHAP has  $\mathcal{O}(LN^2 \log N)$  (Section C.6.1) complexity just to train the model. Therefore, these approaches are unfeasible for highly dimensional problems or large datasets.
- The fastest methods are those which ignore the fitness function ( $\chi^2$  and Mutual Information). The fastest method that uses fitness information is Dominance-based linkage. Both this pre-processing step and the linkage calculation are expected to have  $\mathcal{O}(NL)$  complexity, which matches the observed results.

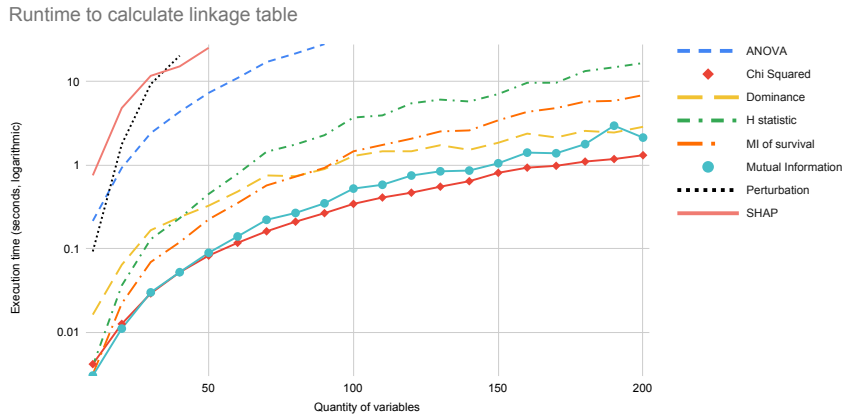


Figure 7.5: Plot showing how the runtime of each linkage method grows as the number of variables increases. The size of the dataset was kept constant at 20000.

### 7.5.7 Conclusion

To summarise the findings, in relation to the research questions posed:

**LRQ1 Which linkage method performs best in general?** Arguably Perturbation, but that is at the cost of new function evaluations. SHAP XGBoost is the best approach which does not involve new fitness function evaluations.

**LRQ2 Does using data from different metaheuristics affect linkage learning?** Although Perturbation performs best on uniformly random solutions, other methods perform better on exploitative metaheuristics.

| Linkage Method: Chi Squared |      |       |       |       |  |
|-----------------------------|------|-------|-------|-------|--|
| Metaheuristic Budget        | 5000 | 10000 | 15000 | 20000 |  |
| Var Count                   |      |       |       |       |  |
| 10                          | 0.00 | 0.00  | 0.00  | 0.00  |  |
| 20                          | 0.01 | 0.01  | 0.01  | 0.01  |  |
| 30                          | 0.03 | 0.03  | 0.04  | 0.03  |  |
| 40                          | 0.06 | 0.05  | 0.06  | 0.05  |  |
| 50                          | 0.08 | 0.08  | 0.08  | 0.09  |  |
| 60                          | 0.12 | 0.12  | 0.12  | 0.12  |  |
| 70                          | 0.19 | 0.16  | 0.16  | 0.16  |  |
| 80                          | 0.21 | 0.22  | 0.21  | 0.22  |  |
| 90                          | 0.29 | 0.27  | 0.29  | 0.29  |  |
| 100                         | 0.33 | 0.33  | 0.33  | 0.48  |  |
| 110                         | 0.40 | 0.40  | 0.40  | 0.40  |  |
| 120                         | 0.47 | 0.49  | 0.47  | 0.47  |  |
| 130                         | 0.55 | 0.56  | 0.56  | 0.56  |  |
| 140                         | 0.66 | 0.69  | 0.65  | 0.65  |  |
| 150                         | 0.75 | 0.76  | 0.74  | 0.76  |  |
| 160                         | 0.86 | 0.87  | 0.86  | 0.86  |  |
| 170                         | 0.97 | 0.98  | 0.96  | 1.00  |  |
| 180                         | 1.06 | 1.07  | 1.09  | 1.11  |  |
| 190                         | 1.27 | 1.21  | 1.20  | 1.20  |  |
| 200                         | 1.38 | 1.32  | 1.32  | 1.44  |  |

| Linkage Method: Mutual Information |      |       |       |       |  |
|------------------------------------|------|-------|-------|-------|--|
| Metaheuristic Budget               | 5000 | 10000 | 15000 | 20000 |  |
| Var Count                          |      |       |       |       |  |
| 10                                 | 0.00 | 0.00  | 0.00  | 0.00  |  |
| 20                                 | 0.01 | 0.01  | 0.01  | 0.01  |  |
| 30                                 | 0.03 | 0.03  | 0.03  | 0.03  |  |
| 40                                 | 0.05 | 0.05  | 0.07  | 0.06  |  |
| 50                                 | 0.08 | 0.10  | 0.14  | 0.11  |  |
| 60                                 | 0.13 | 0.16  | 0.17  | 0.16  |  |
| 70                                 | 0.19 | 0.21  | 0.22  | 0.31  |  |
| 80                                 | 0.24 | 0.30  | 0.34  | 0.37  |  |
| 90                                 | 0.34 | 0.36  | 0.44  | 0.50  |  |
| 100                                | 0.38 | 0.45  | 0.52  | 0.60  |  |
| 110                                | 0.46 | 0.56  | 0.71  | 0.67  |  |
| 120                                | 0.52 | 0.77  | 0.74  | 1.01  |  |
| 130                                | 0.67 | 0.93  | 1.03  | 1.10  |  |
| 140                                | 0.79 | 1.01  | 1.18  | 1.32  |  |
| 150                                | 0.90 | 1.34  | 1.29  | 1.58  |  |
| 160                                | 1.05 | 1.31  | 1.74  | 2.28  |  |
| 170                                | 1.17 | 1.49  | 1.93  | 2.12  |  |
| 180                                | 1.33 | 1.65  | 2.02  | 2.85  |  |
| 190                                | 1.36 | 2.09  | 2.21  | 2.83  |  |
| 200                                | 1.68 | 2.11  | 2.70  | 2.90  |  |

| Linkage Method: Dominance |      |       |       |       |  |
|---------------------------|------|-------|-------|-------|--|
| Metaheuristic Budget      | 5000 | 10000 | 15000 | 20000 |  |
| Var Count                 |      |       |       |       |  |
| 10                        | 0.02 | 0.02  | 0.02  | 0.02  |  |
| 20                        | 0.06 | 0.07  | 0.07  | 0.09  |  |
| 30                        | 0.11 | 0.14  | 0.21  | 0.19  |  |
| 40                        | 0.35 | 0.25  | 0.27  | 0.29  |  |
| 50                        | 0.27 | 0.41  | 0.62  | 0.54  |  |
| 60                        | 0.52 | 0.50  | 0.82  | 0.78  |  |
| 70                        | 0.45 | 0.65  | 0.85  | 1.21  |  |
| 80                        | 0.66 | 0.80  | 1.08  | 1.29  |  |
| 90                        | 1.05 | 0.94  | 1.36  | 2.13  |  |
| 100                       | 0.78 | 1.12  | 1.40  | 1.70  |  |
| 110                       | 0.76 | 1.24  | 1.86  | 1.77  |  |
| 120                       | 0.82 | 1.64  | 1.68  | 2.30  |  |
| 130                       | 1.14 | 1.69  | 1.99  | 2.47  |  |
| 140                       | 1.48 | 1.90  | 2.29  | 2.87  |  |
| 150                       | 1.53 | 2.15  | 2.46  | 3.33  |  |
| 160                       | 1.63 | 2.24  | 2.62  | 3.86  |  |
| 170                       | 2.36 | 2.42  | 3.18  | 3.49  |  |
| 180                       | 1.65 | 2.52  | 2.87  | 4.00  |  |
| 190                       | 1.65 | 2.68  | 3.37  | 4.38  |  |
| 200                       | 2.06 | 2.63  | 3.51  | 4.46  |  |

| Linkage Method: MI of survival |      |       |       |       |  |
|--------------------------------|------|-------|-------|-------|--|
| Metaheuristic Budget           | 5000 | 10000 | 15000 | 20000 |  |
| Var Count                      |      |       |       |       |  |
| 10                             | 0.00 | 0.00  | 0.00  | 0.00  |  |
| 20                             | 0.02 | 0.02  | 0.02  | 0.03  |  |
| 30                             | 0.05 | 0.06  | 0.06  | 0.08  |  |
| 40                             | 0.12 | 0.12  | 0.12  | 0.15  |  |
| 50                             | 0.19 | 0.26  | 0.28  | 0.30  |  |
| 60                             | 0.32 | 0.39  | 0.46  | 0.44  |  |
| 70                             | 0.45 | 0.57  | 0.58  | 0.99  |  |
| 80                             | 0.63 | 0.77  | 0.93  | 1.07  |  |
| 90                             | 0.81 | 0.96  | 1.26  | 1.45  |  |
| 100                            | 1.07 | 1.28  | 1.44  | 1.85  |  |
| 110                            | 1.20 | 1.58  | 2.17  | 1.97  |  |
| 120                            | 1.44 | 2.28  | 2.20  | 3.15  |  |
| 130                            | 1.95 | 2.75  | 2.96  | 3.30  |  |
| 140                            | 2.42 | 3.13  | 3.57  | 4.47  |  |
| 150                            | 2.68 | 3.53  | 3.87  | 5.58  |  |
| 160                            | 3.21 | 4.41  | 4.93  | 6.61  |  |
| 170                            | 3.90 | 4.69  | 6.55  | 6.88  |  |
| 180                            | 4.54 | 5.20  | 8.18  | 8.42  |  |
| 190                            | 4.32 | 6.04  | 7.19  | 10.00 |  |
| 200                            | 5.51 | 6.70  | 8.62  | 10.16 |  |

| Linkage Method: H statistic |      |       |       |       |  |
|-----------------------------|------|-------|-------|-------|--|
| Metaheuristic Budget        | 5000 | 10000 | 15000 | 20000 |  |
| Var Count                   |      |       |       |       |  |
| 10                          | 0.00 | 0.00  | 0.00  | 0.00  |  |
| 20                          | 0.03 | 0.04  | 0.03  | 0.04  |  |
| 30                          | 0.08 | 0.11  | 0.14  | 0.15  |  |
| 40                          | 0.23 | 0.25  | 0.30  | 0.30  |  |
| 50                          | 0.35 | 0.51  | 0.69  | 0.70  |  |
| 60                          | 0.63 | 0.95  | 1.10  | 1.09  |  |
| 70                          | 0.98 | 1.32  | 1.41  | 2.56  |  |
| 80                          | 1.27 | 1.83  | 2.25  | 2.64  |  |
| 90                          | 1.83 | 2.34  | 3.50  | 3.88  |  |
| 100                         | 1.99 | 2.73  | 3.77  | 4.21  |  |
| 110                         | 2.37 | 3.52  | 5.13  | 4.69  |  |
| 120                         | 2.59 | 5.28  | 5.01  | 7.15  |  |
| 130                         | 3.67 | 5.98  | 7.32  | 7.89  |  |
| 140                         | 4.29 | 6.70  | 8.59  | 10.22 |  |
| 150                         | 5.07 | 7.58  | 10.08 | 12.45 |  |
| 160                         | 6.09 | 8.15  | 11.34 | 16.74 |  |
| 170                         | 6.77 | 10.33 | 14.73 | 16.07 |  |
| 180                         | 7.30 | 11.06 | 15.30 | 20.20 |  |
| 190                         | 7.17 | 13.15 | 16.93 | 21.88 |  |
| 200                         | 9.42 | 14.17 | 18.76 | 22.67 |  |

| Linkage Method: ANOVA |       |       |       |       |  |
|-----------------------|-------|-------|-------|-------|--|
| Metaheuristic Budget  | 5000  | 10000 | 15000 | 20000 |  |
| Var Count             |       |       |       |       |  |
| 10                    | 0.20  | 0.20  | 0.20  | 0.19  |  |
| 20                    | 0.91  | 0.93  | 0.93  | 0.99  |  |
| 30                    | 2.15  | 2.35  | 3.75  | 2.60  |  |
| 40                    | 4.33  | 4.81  | 4.91  | 4.72  |  |
| 50                    | 6.70  | 7.85  | 8.51  | 9.04  |  |
| 60                    | 10.79 | 11.59 | 13.53 | 13.07 |  |
| 70                    | 14.74 | 17.41 | 17.58 | 23.97 |  |
| 80                    | 20.67 | 22.91 | 24.93 | 28.77 |  |
| 90                    | 25.59 | 28.97 | X     | X     |  |
| 100                   | X     | X     | X     | X     |  |
| 110                   | X     | X     | X     | X     |  |
| 120                   | X     | X     | X     | X     |  |
| 130                   | X     | X     | X     | X     |  |
| 140                   | X     | X     | X     | X     |  |
| 150                   | X     | X     | X     | X     |  |
| 160                   | X     | X     | X     | X     |  |
| 170                   | X     | X     | X     | X     |  |
| 180                   | X     | X     | X     | X     |  |
| 190                   | X     | X     | X     | X     |  |
| 200                   | X     | X     | X     | X     |  |

| Linkage Method: SHAP |       |       |       |       |  |
|----------------------|-------|-------|-------|-------|--|
| Metaheuristic Budget | 5000  | 10000 | 15000 | 20000 |  |
| Var Count            |       |       |       |       |  |
| 10                   | 0.82  | 0.91  | 0.83  | 0.76  |  |
| 20                   | 3.97  | 4.77  | 4.37  | 5.86  |  |
| 30                   | 6.88  | 10.28 | 13.35 | 15.94 |  |
| 40                   | 16.62 | 18.57 | 22.42 | 21.48 |  |
| 50                   | 18.79 | X     | X     | X     |  |
| 60                   | 28.39 | X     | X     | X     |  |
| 70                   | X     | X     | X     | X     |  |
| 80                   | X     | X     | X     | X     |  |
| 90                   | X     | X     | X     | X     |  |
| 100                  | X     | X     | X     | X     |  |
| 110                  | X     | X     | X     | X     |  |
| 120                  | X     | X     | X     | X     |  |
| 130                  | X     | X     | X     | X     |  |
| 140                  | X     | X     | X     | X     |  |
| 150                  | X     | X     | X     | X     |  |
| 160                  | X     | X     | X     | X     |  |
| 170                  | X     | X     | X     | X     |  |
| 180                  | X     | X     | X     | X     |  |
| 190                  | X     | X     | X     | X     |  |
| 200                  | X     | X     | X     | X     |  |

| Linkage Method: Perturbation |       |       |       |       |  |
|------------------------------|-------|-------|-------|-------|--|
| Metaheuristic Budget         | 5000  | 10000 | 15000 | 20000 |  |
| Var Count                    |       |       |       |       |  |
| 10                           | 0.09  | 0.12  | 0.13  | 0.11  |  |
| 20                           | 1.52  | 1.79  | 1.82  | 2.24  |  |
| 30                           | 5.21  | 8.67  | 8.99  | 10.49 |  |
| 40                           | 19.41 | 22.04 | 25.88 | 27.80 |  |
| 50                           | X     | X     | X     | X     |  |
| 60                           | X     | X     | X     | X     |  |
| 70                           | X     | X     | X     | X     |  |
| 80                           | X     | X     | X     | X     |  |
| 90                           | X     | X     | X     | X     |  |
| 100                          | X     | X     | X     | X     |  |
| 110                          | X     | X     | X     | X     |  |
| 120                          | X     | X     | X     | X     |  |
| 130                          | X     | X     | X     | X     |  |
| 140                          | X     | X     | X     | X     |  |
| 150                          | X     | X     | X     | X     |  |
| 160                          | X     | X     | X     | X     |  |
| 170                          | X     | X     | X     | X     |  |
| 180                          | X     | X     | X     | X     |  |
| 190                          | X     | X     | X     | X     |  |
| 200                          | X     | X     | X     | X     |  |

Figure 7.6: Tables showing the average run time in seconds for each linkage learning method, varying the quantity of variables and the size of the  $P_{\text{Ref}}$  used. Darker colours represent longer times. Crosses represent situations where the algorithm took longer than 30 seconds.

**LRQ3 Which benchmark problems are hard for linkage learning?** Sigmoid Trap-k, Parity and Sigmoid Parity. This might be due to the complex fitness function which involves higher-order cliques.

**LRQ4 Which linkage methods are susceptible to non-linearity?** Although a decrease in performance is observed when introducing non-linearity into a problem, this effect is not as significant as expected. An exception is Trap-k, since most methods perform poorly on Sigmoid Trap-k.

**LRQ5 How does the time complexity of different linkage methods scale with respect to the quantity of variables and the quantity of solutions?** Perturbation, ANOVA and SHAP scale poorly at higher variable counts and dataset sizes. Methods which ignore the fitness function are the fastest, but dominance-based linkage is on par with them.

Overall, there is no method which is always preferable. Perturbation gives the best results, but it requires a live fitness function. SHAP gives the best results by using a static dataset, but its time complexity makes it unfeasible for larger problems and datasets. **Dominance-based linkage** has the next best predictive accuracy, whilst still being computationally efficient.

A comprehensive set of linkage tables is provided in Chapter D for the viewer to get an intuitive understanding of how each linkage method performs on various benchmark problems and metaheuristics.

Having set the theory for how **PSs** can be obtained, this chapter demonstrates how they can be applied. These examples consist of the main contributions of this thesis, with **PSs** being used for explanations for metaheuristics in Section 8.1 and for a recommender model in Section 8.2. They are also used within a predictive model in Section 8.3.

## 8.1 EXPLAINING METAHEURISTIC SOLUTIONS: USE CASE ON STAFF ROSTERING

The primary function of a **PS** is to produce explanations. An extensive example was shown in Section 5.7 already, so this section will focus on what aspects of **XAI** these explanations relate to. It is important to note that even when **PSs** are used for other purposes - such as predicting fitness - the individual **PSs** can always be post-processed into explanations. Figure 8.5 is an example where a decision tree made from **PSs** is extended with explanations.

### 8.1.1 Summary of methodology

**PSs** explain solutions to metaheuristic problems, by either explaining a single solution or by explaining the set of solutions provided in the  $P_{\text{Ref}}$ .

Consider a scenario in which a user is interested in solving an optimisation problem using a certain metaheuristic. Then the overall process is:

1. Obtain a static dataset of solutions (the  $P_{\text{Ref}}$ ), along with their fitness values, as discussed in Section 6.4. If available, the proxy-fitness function values can be included in this dataset.
2. The user decides if they want a **local** explanation or a **global** explanation. If the explanation is local, they also decide a solution to be explained (Section 5.5).
3. The user decides which objectives will be used to find the patterns, to match their requirements. Some relate to their simplicity (Section 6.8), some relate to their linkage (Section 6.9) and some quantify their effect on fitness (Section 6.7).
4. A **PS** mining search is executed by using the aforementioned objectives (Section 6.2), and a Pareto Front is obtained. The user may cluster these patterns (Section 6.13) or they may select one solution (Section 6.6.3). The user may run the **PS** mining process multiple times to obtain a more comprehensive set of results, using **sequential niching** as shown in Section 6.5.2.4.
5. For a given **PS**, we can find and display likely reasons for its relevance, by adding **descriptors** (Section 6.12). Other statistical information can be provided as well, e.g. proving that solutions containing the pattern have a higher average by showing the result of a Mann-Whitney U test.

### 8.1.2 Methodology

Consider the Staff Rostering problem posed by **BT**, as published by us in [46]. The procedure stated above is followed.

**STEP 1: OBTAINING THE STATIC DATASET** **BT** has developed a specific metaheuristic ((iRoster, [81])) to solve their optimisation problem (Section 2.2.7.1), and its results are analysed.

The original metaheuristic (Simulated Annealing for iRoster, defined in [81]) is applied with little modification, with the only difference being that its evaluated solutions are stored for analysis into a  $P_{\text{Ref}}$ . For this metaheuristic (solving the original problem), the fitness evaluation budget was 10000, using a mutation rate of  $\frac{1}{L}$  with uniform mutation operator (defined in Section 2.3.6.2).

In order to obtain descriptors at the end, our industrial point of contact provided various potential proxy fitness functions, which were considered here:

- Mean number of rota choices per worker
- Rota difference (Hamming Distance, in days) between the assigned rotas
- Number of Saturdays and Sundays with at least one worker assigned
- Skill coverage and overlap between the workers
- Percentage of workers assigned to their preferred rota

**STEP 2: LOCAL OR GLOBAL** One of **BT**'s main aims in explanations is to apply slight modifications to the best solutions produced by the metaheuristic. Therefore, **BT** has a greater interest in **local** explanations and thus these are the focus of this analysis. Global explanations will also be explored for more general insights.

**STEP 3: DECIDE OBJECTIVES** One of the most important objectives is always to keep the patterns simple, for which we have **\*-count simplicity**. The second objective considered is **mean-fitness**, to find patterns associated with high fitness. Finally, because there is a large degree of interaction between the variables of the problem, **atomicity** is included as an objective. The specific variant here is bivariate atomicity based on Mutual Information of Survival (Section 7.4.1).

*(Mutual Information of Survival was used here despite the existence of better linkage methods because at this point in the research process a better atomicity metric had not been found.)*

**STEP 4: PARTIAL SOLUTION MINING** NSGA-II is used to solve the **PS** mining task, which results in a Pareto front of **PSs**. These consist of runs with a **PS** evaluation budget of 5000 (population size of 150, crossover rate = 0.5), executed sequentially using the niching operator, with 5 runs in total. These are then clustered using the criterion in Section 6.13 and then ordered according to the heuristic presented in Section 6.6.3. The top 3 results of this ordering are post-processed and then shown.

The process (mining, clustering, sorting, post-processing) is done twice: once for local explanations of the best solution found, and another time for global explanations. The parameters used were determined by the evaluation shown in Table 6.1, indicating that many small sequential runs are preferable. This quantity of sequential runs was also found to be satisfactory because it is able to explain 80% of the given solutions, which accounts for nearly all of the variables with cardinality greater than 1. Further iterations would be expected to find patterns which are redundant with what had been found already.

**STEP 5: DESCRIPTORS** Using the proxy-fitness functions provided in the first step, **descriptors** can be found for the selected **PSs**. The threshold for these descriptors was set to 10%, meaning that proxy-fitness values which are either in the relative top 10% or bottom 10% were shown.

### 8.1.3 Local Explanations

The program displays the **PSs** contained within the optimum found by iRoster. Each explanation consists of a group of employees and their rota assignments (**PS**), and its descriptors accompanied by their percentile (see Section 6.12). Each **PS** has an “effect on fitness” representing the mean difference in fitness between solutions which contain the pattern and those that do not – a negative value is preferred in this problem.

**Example A** presents a group of workers whose chosen rotas appear to be convenient for covering Sundays and that are also the workers’ preference. Moreover, their rotas are different from each other by nearly 24 days on average. Their interaction is also explained by the similarity of their skills.

**Example B** is a smaller group of people, and this is reflected in a smaller effect on fitness. As opposed to Example A, according to the percentile of the descriptors, these workers have very similar rotas. It appears that this group is better suited to cover Saturdays rather than Sundays, as opposed to Example A.

Finally, **Example C** contains another configuration, but with fewer descriptors than the previous examples. This group specialises in covering Sundays but not Saturdays, and their skills are similar (i.e. there are overlaps).

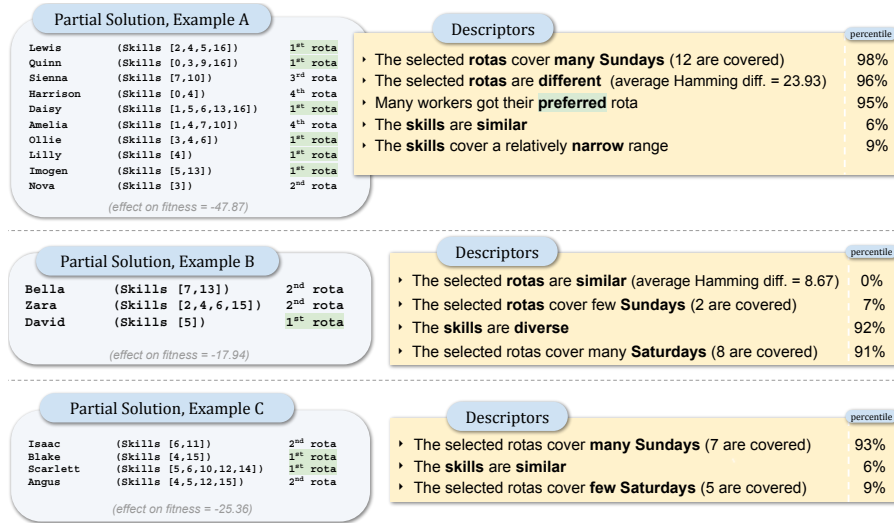


Figure 8.1: Three explanations produced by the system. The patterns are presented in the blue regions, whereas the descriptors are presented in the yellow regions. The percentile of each descriptor is also presented.

#### 8.1.4 Utility of local explanations

In case a worker would like to know why they were assigned to a certain rota, these explanations can provide a lot of insight. If, for example, Sienna wanted to know why she was assigned to her 3<sup>rd</sup> rota, she would be presented with Example A and any other explanations pertaining to her. From Example A she would infer that her rota choice needs to work well with the workers in that group because they have similar skills, and that her rota should be very different from them.

#### 8.1.5 Global Explanations

Additional insights can be gained by analysing all of the PSs obtained by the PS mining algorithm through a **global search**. 110 patterns were obtained in total.

The first result is the frequency with which the workers appear in PSs. The frequency of certain employees being high is an indication that they interact with more groups, and thus that their rota choices are particularly important.

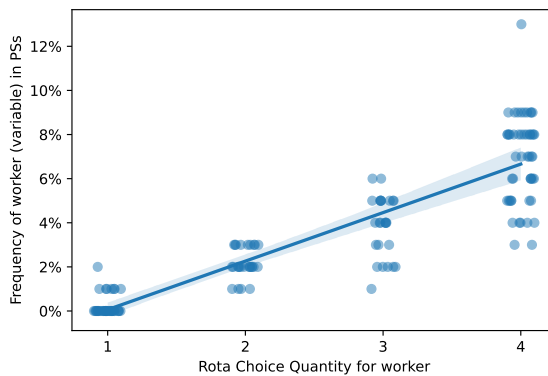


Figure 8.2: A scatter-plot to show the correlation between the frequency of workers, each corresponding to a variable, appearing in PSs, and the quantity of rota choices which are available to them. A light horizontal jitter was applied.

Workers with more rota choices available were found to appear more often in PSs, which was tested using Pearson’s correlation. The approximated coefficient is 0.88, with p-value  $\ll 0.001$  and it explains  $R^2 = 78\%$  of variance. This relationship - shown in Figure 8.2) **suggests that employees with more rota choices have a greater impact on the fitness**, possibly due to increased tunability, which was observed in [96] as well.

It was also found that workers with skills 5 and/or 15 tend to appear more often. We tested this using a  $\chi^2$  test, to check if each skill appears in PSs with the distribution expected, and the p-values are 0.063 and 0.024 respectively. It is unclear why these skills in particular are highly valued, but further investigation found that workers with these skills produced more optimised weekends.

#### 8.1.6 Utility of global explanations

The information obtained in the local and global explanations can be used by decision makers to decide what changes to implement within their workforce in order to improve the settings of their Staff Rostering task. For example, if a manager wanted to improve the importance of the less critical workers, they could train them in the skills found to be important in the global explanations. Additionally, knowing which skills are important could be used to guide the hiring process and explain to candidates why they were turned down and what they could do to improve their chances in the future.

Once the workers are hired, the managers will be aware that having multiple rotas is also beneficial, and therefore they will make many rota choices available to the new hires. To decide these rotas, they might look at the local explanations of the current solutions and find groups of workers that already appear to be interacting due to those skills. Looking at the rotas of those workers and at descriptors relating to whether they are similar/different, the manager can propose rotas that are likely to harmonise with the existing workforce.

#### 8.1.7 Evaluation of metaheuristic explanations

The explanations shown above, which were produced via PS mining, do not show whether our methodology is able to recover *true* explanations. This evaluation presents a challenge: it is not knowable what the “true” explanation is for a real-world instance of the Staff Rostering problem. Therefore, it is first necessary to design problem instances so that they are explained by certain explanations.

For this purpose, two problem instances were created: SR-RR and SR-GC. These are Staff Rostering instances which behave like RoyalRoad and Graph Colouring. Their full definition is discussed in Chapter E, and in summary:

- SR-RR is explained by 5 separate cliques of workers, each having 4 employees. Each employee has 2 possible rotas. Each clique contributes positively to fitness only if they are *all* assigned to their second rota.
- SR-GC is explained by 5 separate cliques of workers, each having 3 employees. Each employee has 3 possible rotas. A clique contributes positively to fitness if all its employees are assigned to *different* rotas.

The first problem is used to represent “all or nothing” scenarios, whereas the second kind is used to represent a scenario with higher-order relationships, which we expect to be hard to capture through PS (since there are 6 different ways in which each clique is “optimal”).

The problems are solved using iRoster (i.e. Simulated Annealing with uniform mutation rate, to capture to BT’s use case scenario) with an evaluation budget of 10000, producing a  $P_{\text{Ref}}$  of 10000 solutions.

The table presented earlier in the thesis, Table 6.1 contains this evaluation. The evaluation also intends to investigate which multi-objective metaheuristic performs best, and with what parameters. All the combinations have the same fitness evaluation budget of 10000. The possible parameters are:

- Many small low-budget runs, each with budget of 1000 and a population of 50;
- Some medium-budget runs, each with budget of 5000 and a population of 100;

- A single run using all the budget, with a population of 200;

Here each run is a sequential niching run in decision space, as defined in Section 6.5.2.4.

A **PS** mining run is evaluated by checking if *all* the “ground truth” **PS**s are present in the resulting final set of **PS**s. Table 6.1 is a table indicating the percentage of **PS** mining runs which were successful. If a run finds *some* but not all the ground truth patterns, then it is still considered unsuccessful.

The results indicate the following results:

- NSGA-II is the best performing multi-objective method;
- Using many small runs (budget of 1000 per run) yields the highest probability of finding all the patterns.
- SR-RR **PS** mining runs reach a 93% success rate, whereas SR-GC reach a 68% success rate at best. This indicates that, as expected, higher order relationships between variables are harder to capture via **PS mining!** (**PS mining!**).

The main outcome of this evaluation is that our methodology is able to capture *all* ground truth explanations, provided that the adequate parameters are provided.

## 8.2 EXPLANATIONS FOR RECOMMENDER MODELS

**PS**s are not limited to datasets relating to metaheuristics, but can be applied to any dataset where combinatorial vectors are present. This was relevant when collaborating with Recommender Model researchers who were looking for explanations, which then led to the development of **PS4XRS** [43].

In summary: a recommender model is evaluated on real users, which results in a large and complex dataset. This dataset can be processed to be closer in format to a  $P_{Ref}$ , which allows us to apply **PS** mining. The definition of **PS** needs to be changed slightly for this context in order to better suit the data, and useful **PS** are obtained and post-processed into explanations.

The patterns we find describe a user’s interaction history, and they are associated with higher predictive performance from the recommender model. An example of these explanations is given in Figure 8.3, where it was found that the model is 10% more accurate on users who interacted with certain categories of books. This information is useful to stakeholders, for example in knowing when it is worth displaying advertisements.

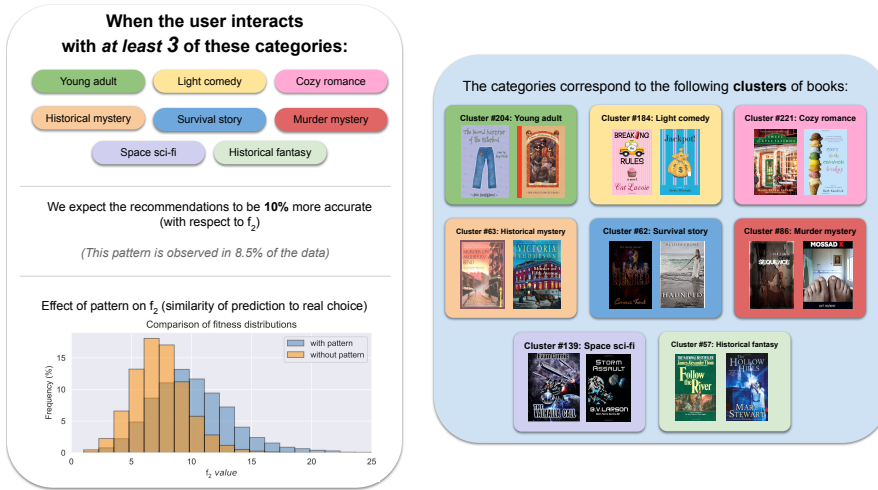


Figure 8.3: Example of explanation produced for PS4XRS.

**RECOMMENDER SYSTEMS** Recommender Systems are **ML** models designed for the selection or creation of content towards a user, informed by their past activities [188]. One of the most challenging aspects of these models is their evaluation, since their performance ought to be measured via large-scale studies involving human users.

Additionally, the monetary component makes the performance of these models critical, which motivates the need for **explainability**. In the case discussed here, knowing **when**

**the model performs well** can provide essential information towards customer insights and marketing.

### 8.2.1 *Recommender model*

The recommender model developed in [43] by our collaborators makes use of training data which is in the form of a network, where nodes represent users and items within an online book retail website, and these are connected if there is an interaction between them. An interaction consists of a user clicking an item, reviewing it, or purchasing it.

The large quantity of items and users makes the process of learning from this data complex, but it is possible to learn **embedded vectors**. The main purpose of these embeddings is to measure the similarity between items, and to measure the compatibility between users and items. **Cosine similarity** is a simple mathematical operation used to measure these similarities. This architecture is how the **LightGCN model** is implemented [125].

**SOURCE DATA** Once the model is trained, it can be deployed on human users, but in this case a proxy evaluation method was used. The YooChoose Amazon Book dataset [245] is a large collection of user interaction data, used for this study. The dataset is partitioned in 3 parts: one for training the model, one for testing the Recommender models, and one for testing the explanations.

Once the model has been trained on the first partition, it is applied to the second by predicting which item the user will purchase next. This is then compared against the ground truth, which gives a “quality” of prediction score for each user. Each user can be represented as a sequence of boolean variables, where each variable  $i$  indicates any interaction with book  $i$ . In this sense, each user looks like the solution to a binary optimisation problem. The “fitness” for these solutions is the quality of the recommendation prediction for that specific user. This information follows the format of a  $P_{\text{Ref}}$ , although some adjustments will be required in order for **PS** mining to be immediately applicable.

**CHALLENGES** This research posed the following challenges:

- **Too many variables:** the dataset discussed above contained thousands of books, which results in solutions with thousands of variables. Due to many reasons (curse of dimensionality for **PS** mining, patterns that need to be small), this is not usable.
- **Sparse dataset:** a user who interacts with 4 items in total will be associated with a solution with 4 variables being set to 1, and the rest being set to 0. In general, most users bought fewer than 5 items, which causes most of the dataset to be composed of zeros. For statistical reasons, any **PS** with more than 2 ones is unlikely to have any matching solutions, which is proven in Section 8.2.2.1.
- **The format of patterns:** if a solution is a binary vector, where 1 = interaction with book, and 0 = non-interaction, a **PS** like (\* \* 1 0) represents “users who interacted with book 2 and did NOT interact with book 3”. For stakeholders, non-interactions are not informative and harder to understand, therefore **PS** should not include these. In simpler terms, **PSs** should not contain zeros.
- **Different objectives:** although in an optimisation context it does not matter whether a pattern is “common” or “rare”, in this case it does matter. If a pattern identifies a kind of scenario where the recommender model performs well, a stakeholder would want this to be a common scenario. Moreover, the meaning of linkage between variables is taken here to be the similarity between store items.

### 8.2.2 *Resolution of challenges*

Here we resolve the issues described above. The core issue is the dataset being chronically sparse, which was resolved via clustering, custom **PSs** and custom search operators. Other than that, the rest of the methodology remained the same, which is a testament to how flexible **PS** mining can be. A summary of the overall methodology is shown in Figure 8.4.

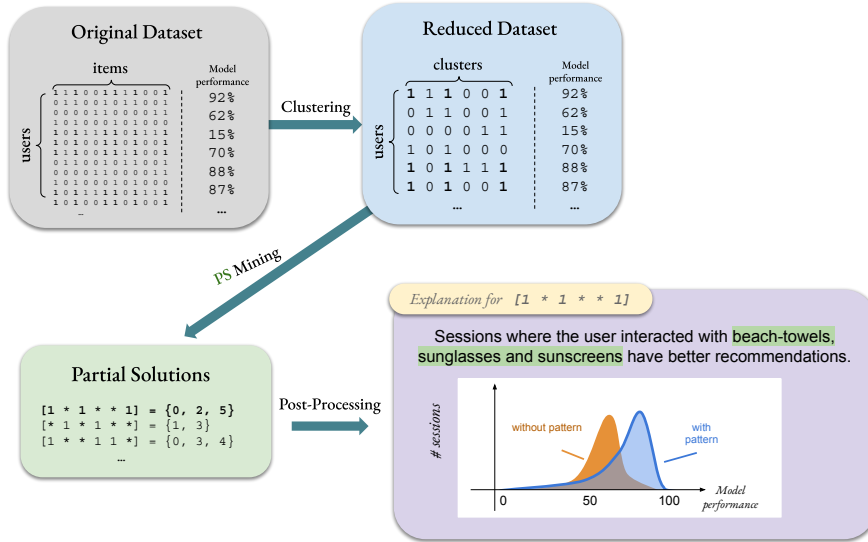


Figure 8.4: Summary of methodology used for PS4XRS. The initial dataset is reduced in dimensionality using clustering, and this is then treated as a  $P_{\text{Ref}}$ . After mining PSs from this dataset, they are post-processed into explanations.

#### 8.2.2.1 Clustering of variables

Due to the quantity of retail items present in the dataset, it is necessary to reduce the dimensionality of the solutions. **Clustering** is applicable here, because the model offers a way to measure similarity between items.

If one wanted to work with 100 variables, the required procedure is to cluster all the retail items using **cosine similarity** as their distance metric and then set the desired cluster quantity to 100. After obtaining the clusters, the columns corresponding to each item within a cluster are aggregated into one using logical **or**. Each cluster becomes one variable.

k-means clustering with random initialisation was used by our collaborators Balcer and Lipinski in [43] in order to provide a “uniform” set of items rather than focusing on the most purchased items [43]. Our collaborators decided to make 250 clusters, which is justified later in Section 8.2.2.2.

Clustering has the added advantage of alleviating the issue of sparsity, to an extent. Fewer clusters result in fewer zeros, as Table 8.1 shows the relationship between these two properties. In summary, each variable in the solutions now represents a *cluster* of books rather than individual books. The “genre” of each cluster was easily determined by looking through a sample of books for each cluster.

Table 8.1: Data sparsity for different quantities of clusters. For every cluster of products, there is a certain probability that a given user has interacted with it.

| # Clusters | Probability of interaction |
|------------|----------------------------|
| 20         | 47%                        |
| 50         | 25%                        |
| 75         | 19%                        |
| 100        | 16%                        |
| 125        | 13%                        |
| 250        | 8%                         |
| 500        | 4%                         |
| 1000       | 2%                         |

### 8.2.2.2 Selecting the dataset

The dataset to be used for Recommender System Partial Solutions (RSPS) mining was to be created from two parameters: how many clusters, and which “model quality metric” to use. The model quality metric is not relevant to this work, and all the reader needs to know is that there were 6 ways of quantifying how well the recommender model predicted the behaviour of each user.

Initially, it was unclear whether the clustering steps would allow for the data to still retain any pattern, and a simple test was devised: if a machine learning model is able to generalise on a certain dataset, it meant that it contained enough patterns.

For the many possible obtainable datasets (cluster quantity  $\in \{20, 50, 100, 125, 250, 500, 1000\}$ , model fitness  $\in \{f_1, f_2, \dots, f_6\}$ ) we trained and tested machine learning models with an 80%-20% split respectively. We used both a traditional linear model with L2 regularisation and an MLP with 1 hidden layer of 30 neurons and measured the predictive performance using  $R^2$  to be scale invariant.

The first 20 rows are shown in Table 8.2. The main insights are:

1.  $f_2$  occupies all of the top positions, indicating that it is the most informative fitness metric,
2. more clusters lead to more information retained

**Using this information, it was decided to use  $f_2$  as the model evaluation metric, and to use 250 clusters as there is only a marginal improvement between 250 and higher quantities.**

$f_2$  is a metric which measures how closely the recommender model was able to predict a user’s last product interacted with. It is the dot-product between the real item’s embedded vector, and that of the predicted item [43].

This procedure may be used beyond PS4XRS and more generally to determine if a dataset is compatible with PS mining.

Table 8.2: Table showing how much “information” each dataset retains when using certain clustering quantities and model evaluation methods. A model (either a linear model (LR) or an MLP) was trained and evaluated on the dataset, and the  $R^2$  value is reported. These rows are ordered by  $R^2$  descending.

| Model | Genome Size | Fitness col | $R^2$ |
|-------|-------------|-------------|-------|
| LR    | 1000        | $f_2$       | 0.336 |
| LR    | 500         | $f_2$       | 0.33  |
| LR    | 250         | $f_2$       | 0.326 |
| LR    | 100         | $f_2$       | 0.282 |
| MLP   | 20          | $f_2$       | 0.244 |
| LR    | 50          | $f_2$       | 0.234 |
| MLP   | 50          | $f_2$       | 0.222 |
| LR    | 20          | $f_2$       | 0.187 |
| MLP   | 100         | $f_2$       | 0.086 |
| LR    | 250         | $f_3$       | 0.084 |
| LR    | 100         | $f_3$       | 0.076 |
| LR    | 1000        | $f_3$       | 0.072 |
| LR    | 500         | $f_3$       | 0.069 |
| LR    | 50          | $f_3$       | 0.067 |
| LR    | 500         | $f_4$       | 0.059 |
| LR    | 20          | $f_3$       | 0.058 |
| LR    | 250         | $f_4$       | 0.058 |
| MLP   | 20          | $f_3$       | 0.053 |
| LR    | 250         | $f_5$       | 0.038 |

### 8.2.2.3 Patterns without zeros

Our collaborators indicated that potential users would prefer for the explanations not to contain any non-interactions, which in practice means to avoid zeros in the patterns for explanations.

Now that some figures are available for the sparsity, the reader can understand how knowing that a certain variable has value 0 (i.e., the cluster of items has not been interacted with) has little information content.

Therefore, the search for **PSs** was modified (by myself) to not allow zeros within patterns. If a pattern is a sequence of mostly \* and ones, then it is much more efficient to simply store *which* variables have the fixed value of one; thus the representation of patterns here is as a set of integers (as shown in Figure 8.4).

### 8.2.2.4 Embracing sparsity by modifying Partial Solutions

The sparsity of the dataset is at odds with the core idea behind **PSs**: they need to match enough solutions in order to make conclusions about the pattern itself.

Consider the case where the items have been put into 250 clusters, and we are analysing a pattern with 3 fixed variables. Even this very simple pattern will match very few solutions, since each item has an 8% chance of being interacted with ( $0.08^3 \approx 0.0005 = 0.05\%$  of users). Instead of considering patterns where “all fixed variables need to be matched”, we consider patterns with more relaxed requirements, such as “solutions where at least 3 of these variables are 1: {34, 45, 70}”. To implement this, a **threshold parameter**  $t$  is included within the **PS**, leading to a novel kind of **PS**: the **RSPS**. This parameter may be fixed during mining, or it may be itself subject to evolution - both options were considered during evaluation.

Formally, an **RSPS** is defined as  $(r^\psi, t)$ , where:

$$r^\psi = \{r_1^\psi, r_2^\psi, \dots\}, \quad r_i^\psi \in [1..L] \quad (8.1)$$

$$t \in \mathbb{N} \quad (8.2)$$

The pattern  $r^\psi$  **matches** a solution  $s$  if

$$\left| \{i \in [1..L] \mid s_{r_i^\psi} = 1\} \right| \geq t \quad (8.3)$$

Where  $L$  is the quantity of clusters being considered,  $r^\psi$  is the pattern (a set of indices),  $s_{r_i^\psi} = 1$  checks if the item cluster at index  $r_i^\psi$  has been interacted with, and  $t$  is the integer threshold set by the **RSPS** for how many clusters specified by the pattern need to have been interacted with in a user’s interaction history.

### 8.2.2.5 Custom Objectives

The following objectives were used for the **RSPS** mining task:

- **Simplicity: \*-count simplicity**: the patterns need to be simple to be understandable
- **Mean-fitness**: the patterns need to be associated with higher recommender system performance
- **Atomicity**: bivariate, based on linkage information that can be obtained from the model (cosine similarity)
- **Sample quantity**: counts how many solutions  $P_{\text{Ref}}$  match the pattern, which is equivalent to the quantity of matched users

The last two metrics are unique to this work. The simplest one to explain is **sample quantity**: since the explanations are used to explain general patterns within the model and the users, it is useful for the stakeholders to obtain common patterns.

The new atomicity metric is bivariate atomicity, but the source of linkage information comes from the item embeddings. The linkage between two variables is the cosine similarity between the centroids of the clusters they correspond to. This linkage information is relevant to implementing custom search operators.

### 8.2.2.6 Custom search operators

The search operators used to evolve patterns – sampling, mutation, and crossover – were customised to improve the search process by using the information contained within the recommender model.

**SAMPLING** Some clusters were interacted with more often than others. Let  $p_i$  indicate the probability that cluster  $i$  was interacted with, and this allows the implementation of a sampling operator that is essentially uniform sampling with custom probabilities.

---

#### Algorithm 16 RS-Sampling

---

```

function SAMPLE
   $s^\psi \leftarrow \emptyset$ 
  for  $i \leftarrow 0$  to  $N - 1$  do
    if  $\text{random}() < p_i$  then
       $s^\psi \leftarrow s^\psi \cup \{i\}$ 
    end if
  end for
  return  $s^\psi$ 
end function

```

---

In conventional **PS**-initialisation, a pattern is initially “empty”, and following a geometric distribution they are filled with values at random indices (Section 6.5.2.1). **RS-sampling** (Algorithm 16) will use the probability of each cluster being interacted with (here it will be  $p_i$ ) to determine how often each item appears in **PSs** initially (Algorithm 16).

**MUTATION** To guide modifications based on cosine similarity, we use a Markov-like transition model, where the transition probability between clusters  $i$  and  $j$  is based on the cosine similarity of their embedded centroids, and constitutes element  $M_{ij}$  of a matrix  $M$ . We rescale this matrix (to have values between 0 and 1,  $M_s$ ) and normalise it (sum of each row is 1) to construct a Markov transition matrix  $M_n$ :

$$M_s = \frac{M - \min(M)}{\max(M) - \min(M)}, \quad M_n[i, :] = \frac{M_s[i, :]}{\sum_j M_s[i, j]}$$

Let  $r \in [0, 1]^N$  be the one-hot encoding binary vector of a pattern  $s^\psi$  ( $r_i = 1$  for items in  $s^\psi$ , 0 otherwise). Then  $r' = rM_n$  gives a similarity-weighted probability that each item should appear in the next version of the pattern (as if it were passed through the Markov model defined by  $M_n$ ). We can then “quantise”  $r'$  into a new pattern (i.e., converting from probabilities to a **RSPS**,  $[0, 1]^L \rightarrow \text{RSPS}$ ):

$$\text{Quantise}(r') = \{i \mid \text{random}() < r'_i\}$$

However, it is hard to control the size of the patterns produced by quantising  $r'$ . The average quantity of “successes” (fixed variables) from the quantisation is actually the sum of probabilities in  $r'$  [213], and we can easily rescale  $r'$  to have a desired expected average size. Let this be  $a$ , then

$$r'' = r' \cdot \frac{a}{\sum r'}$$

**RS Mutation** uses these ideas directly, where a mutation is an iteration through the Markov process (Algorithm 17).

---

#### Algorithm 17 RS-Mutation

---

```

function MUTATE( $s^\psi$ )
   $r \leftarrow \text{as\_vector}(s^\psi)$ 
   $r' \leftarrow r \cdot M_n$ 
   $r'' \leftarrow r' \cdot \frac{\text{len}(s^\psi)}{\sum r'}$ 
  return  $\text{Quantise}(r'')$ 
end function

```

---

**CROSSOVER** Traditional **PS** crossover performs uniform crossover (Section 6.5.2.3). In **RS Crossover** the parents are overlapped (we take the union of the items interacted with) and then mutated by passed through the Markov model used in RS-Mutation. The pre-quantised child ( $r'$ ) is normalised by a value ( $\sum \frac{\ell}{r'}$ ) so that its expected **fixed\_count** is the average of the parent sizes (Algorithm 18). The first child is then quantised, and the second child is created such that (1) the union of the children produces the union of the parents, and (2) the intersection of the children is the intersection of the parents.

---

**Algorithm 18** RS-Crossover

---

```

function CROSSOVER( $a^\psi, b^\psi$ )
     $merged \leftarrow a^\psi \cup b^\psi$ 
     $r \leftarrow \text{as\_vector}(merged)$ 
     $r' \leftarrow r \cdot M_n$ 
     $\ell \leftarrow \frac{|a^\psi| + |b^\psi|}{2}$ 
     $r'' \leftarrow r' \cdot \sum \frac{\ell}{r'}$ 
     $s^\psi \leftarrow \text{Quantise}(r'')$ 
     $t^\psi \leftarrow ((a^\psi \cup b^\psi) \setminus s^\psi) \cup (a^\psi \cap b^\psi)$ 
    return  $s^\psi, t^\psi$ 
end function

```

---

### 8.2.3 Evaluation

The explanations produced on this data can be tested for their accuracy, by checking if they **generalise** in the same way that a model would: does the performance of **RSPS** degrade when applied on “unseen” user sessions?.

In general, a **RSPS** explanation asserts that “solutions which contain this pattern have higher fitness than solutions which do not contain the pattern”. We say that a pattern generalises if it has a high  $f_2$  on unseen data.

**PARAMETERS BEING TESTED** The evaluation aims to determine the parameters which lead to the most *trustworthy* results, i.e., those that generalise best. The parameters to be tested are the following:

- The **objectives** are used in **RSPS** mining. To show that all 4 metrics suggested are necessary (Simplicity, User Count, Mean fitness, Atomicity), we compare the performance when each one of them is removed.
- **The threshold parameter  $t$** , which determines how many of the fixed variables need to be matched. This may be “off” (all variables need to be matched), fixed to a number (3, 4 or 5, since the vast majority of users perform 3, 4 or 5 interactions) or make it automatically found by the **GA**. This is achieved via mutation, where  $t$  has a 10% chance of being incremented or decremented.
- **Which search operators to use**. The custom operators suggested in Section 8.2.2.6 are compared against their conventional **PS** counterpart (defined in Section 6.5).

**WHAT IS BEING MEASURED** We measured the following in each **RSPS** mining run:

- The quantity of patterns produced
- The quantity of fixed values in each pattern
- How much of the dataset is matched, by each RSPS, on average
- The  $f_2$  for each **RSPS** on average.

**RUN PARAMETERS** **RSPS** mining is carried out using the methodology described in Chapter 6. NSGA-II search is run with a population size of 100 and an evaluation budget of 10,000, which was found to be sufficient.

Each combination of parameters is executed 100 times. The mutation rate was set to  $1 / \text{fixed\_count}$  (where `fixed_count` is the quantity of fixed variables in a pattern) and the chance of crossover was set to 50%.

**RESULTS** The results are shown in Table 8.3 and are summarised as follows:

1. The **PSs** have **consistent results between train and test data**, meaning that they are generalising and are applicable to unseen data;
2. **RS-Mutation is an improvement over the original operator**, but Sampling and Crossover are less impactful;
3. The four objectives and the threshold are essential to get a range of **PSs** with sufficient coverage and sizes (both in mean and SD);
4. The SD of the pattern size and coverage indicate that there is a wide range of **PSs**.

Using the 4 objectives, with a threshold and using the new operators is the best approach. The threshold itself can be fixed to a certain value or be evolved, and these yield slightly different results which are generally satisfactory.

Comparing the performance of the algorithm when the **4 objectives** are present against when one is missing reveals how essential each of them is. For example, [SCMA,  $t = 3$ , custom operators] results in patterns with 20% user session coverage on average, whereas the same method but without the user session count objective yields  $3\% \pm 1\%$ .

When the **threshold** is not used ( $t = \text{off}$ , all fixed variables need to be matched), only patterns of 1 or 2 variables can be found (see the average pattern size) which have coverage between 3% and 15%. When it is enabled, the pattern sizes are on average 20 (although with a large SD) and they can be much more general.

Another necessary element is the use of the **specialised mutation operator**, since its removal results in much smaller patterns (average size is 5.62) which have less coverage (average is 9%). The sampling and crossover operators are less impactful, since swapping them for the more “traditional” alternative appears to have no effect.

#### 8.2.4 *PS4XRS Conclusions*

The proposed methodology can generate explanations for a Recommender System dataset which give insight that is useful to stakeholders when adopting the system. The explanations cover a wide range, with a clear trade-off between their generality and model prediction improvement, thanks to the use of multi-objective search.

Using the knowledge from the deep learning model encoded in the embedded space allowed us to develop meaningful clustering, evolutionary operators and suitable objectives, and it is likely to be useful for future developments.

| objec-<br>tives | thre-<br>shold | Search<br>operators | average<br>#patterns | pattern size<br>mean $\pm$ SD | metrics on test           |                        | metrics on train          |                        |
|-----------------|----------------|---------------------|----------------------|-------------------------------|---------------------------|------------------------|---------------------------|------------------------|
|                 |                |                     |                      |                               | coverage<br>mean $\pm$ SD | $f_2$<br>mean $\pm$ SD | coverage<br>mean $\pm$ SD | $f_2$<br>mean $\pm$ SD |
| SUMA            | t=3            | RS, RM, RC          | 64.55                | 20.84 $\pm$ 18.64             | 20% $\pm$ 22%             | 29% $\pm$ 4%           | 20% $\pm$ 22%             | 29% $\pm$ 3%           |
|                 | t=4            |                     | 63.62                | 24.80 $\pm$ 17.97             | 16% $\pm$ 17%             | 30% $\pm$ 4%           | 16% $\pm$ 17%             | 30% $\pm$ 4%           |
|                 | t=5            |                     | 66.14                | 29.56 $\pm$ 18.80             | 13% $\pm$ 14%             | 31% $\pm$ 4%           | 13% $\pm$ 14%             | 31% $\pm$ 4%           |
|                 | t=auto         |                     | 72.93                | 26.69 $\pm$ 30.95             | 53% $\pm$ 33%             | 26% $\pm$ 2%           | 53% $\pm$ 33%             | 26% $\pm$ 2%           |
| UMA             | t=3            | RS, RM, RC          | 72.91                | 124.14 $\pm$ 61.17            | 86% $\pm$ 17%             | 24% $\pm$ 1%           | 86% $\pm$ 17%             | 24% $\pm$ 1%           |
| SMA             |                |                     | 23.39                | 6.30 $\pm$ 1.99               | 3% $\pm$ 1%               | 34% $\pm$ 5%           | 3% $\pm$ 1%               | 34% $\pm$ 4%           |
| SUA             |                |                     | 78.09                | 63.62 $\pm$ 46.33             | 60% $\pm$ 33%             | 26% $\pm$ 3%           | 60% $\pm$ 33%             | 26% $\pm$ 2%           |
| SUM             |                |                     | 74.44                | 43.69 $\pm$ 33.00             | 44% $\pm$ 31%             | 27% $\pm$ 3%           | 44% $\pm$ 31%             | 27% $\pm$ 3%           |
| SUMA            |                | TS, RM, RC          | 60.63                | 21.18 $\pm$ 20.14             | 20% $\pm$ 23%             | 29% $\pm$ 4%           | 20% $\pm$ 23%             | 29% $\pm$ 3%           |
|                 |                | RS, TM, RC          | 73.49                | 5.62 $\pm$ 2.37               | 9% $\pm$ 5%               | 32% $\pm$ 3%           | 9% $\pm$ 5%               | 32% $\pm$ 3%           |
|                 |                | RS, RM, TC          | 62.40                | 20.45 $\pm$ 18.47             | 20% $\pm$ 21%             | 29% $\pm$ 4%           | 20% $\pm$ 21%             | 29% $\pm$ 3%           |
| SUMA            | t=off          | RS, RM, RC          | 51.00                | 1.70 $\pm$ 0.50               | 10% $\pm$ 13%             | 30% $\pm$ 4%           | 10% $\pm$ 13%             | 30% $\pm$ 4%           |
| UMA             |                |                     | 43.43                | 1.79 $\pm$ 0.46               | 10% $\pm$ 14%             | 30% $\pm$ 4%           | 10% $\pm$ 14%             | 30% $\pm$ 4%           |
| SMA             |                |                     | 8.18                 | 1.92 $\pm$ 0.41               | 3% $\pm$ 1%               | 32% $\pm$ 7%           | 3% $\pm$ 1%               | 32% $\pm$ 6%           |
| SUA             |                |                     | 8.97                 | 1.89 $\pm$ 0.32               | 15% $\pm$ 22%             | 28% $\pm$ 5%           | 15% $\pm$ 22%             | 28% $\pm$ 4%           |
| SUM             |                | 28.32               | 1.44 $\pm$ 0.54      | 14% $\pm$ 16%                 | 31% $\pm$ 4%              | 14% $\pm$ 16%          | 31% $\pm$ 4%              |                        |
| SUMA            |                | TS, RM, RC          | 49.88                | 1.68 $\pm$ 0.48               | 10% $\pm$ 13%             | 30% $\pm$ 5%           | 10% $\pm$ 13%             | 30% $\pm$ 4%           |
|                 |                | RS, TM, RC          | 68.96                | 2.37 $\pm$ 0.79               | 8% $\pm$ 10%              | 32% $\pm$ 4%           | 8% $\pm$ 10%              | 32% $\pm$ 4%           |
|                 |                | RS, RM, TC          | 50.03                | 1.69 $\pm$ 0.50               | 10% $\pm$ 13%             | 30% $\pm$ 4%           | 10% $\pm$ 13%             | 31% $\pm$ 4%           |
| SMA             |                | t=off               | TS, TM, TC           | 10.43                         | 2.23% $\pm$ 0.7%          | 3% $\pm$ 1%            | 32% $\pm$ 5%              | 3% $\pm$ 1%            |

Table 8.3: Statistics for different ways of obtaining the PSs for N = 250. Here, we test different objectives, match thresholds and search operators. In the objectives, S = Simplicity (pattern size), U = User Count, M = Mean Fitness, A = Atomicity. In threshold,  $t$  = auto means that  $t$  was evolved, and  $t$  = off means that the entire pattern had to match. In the search operators, TS = Traditional Sampling, TM = Traditional Crossover, TC = Traditional Mutation, RS, RM and RC are the customised counterparts. The metrics reported are quantity of patterns per run, pattern size, proportion of sessions matched (coverage), Hit Rate (HR). Means and SD are reported, since having a good spread is in some cases more important than the average itself.

A commonly desired explanation for metaheuristics is one which explains why a solution has a certain fitness value. This can be achieved by using a **transparent regression model** for predicting fitness. Such a model aims to recreate the fitness function, and is therefore a **surrogate**. In the context of metaheuristics this surrogate is typically created to improve efficiency, whereas in **XAI** it is created to simplify an existing model. The work described here, as published in [45], relates to the latter purpose.

This section will show how **PSs** can be used to create a transparent surrogate model of fitness - a decision tree where the decision nodes check if certain patterns are present, as shown in Figure 8.5. The contributions of this section are:

- a novel explainability tool for metaheuristics
- a novel approach to model fitness functions using **PSs**
- an evaluation of our methods for different problems and metaheuristics against existing alternatives

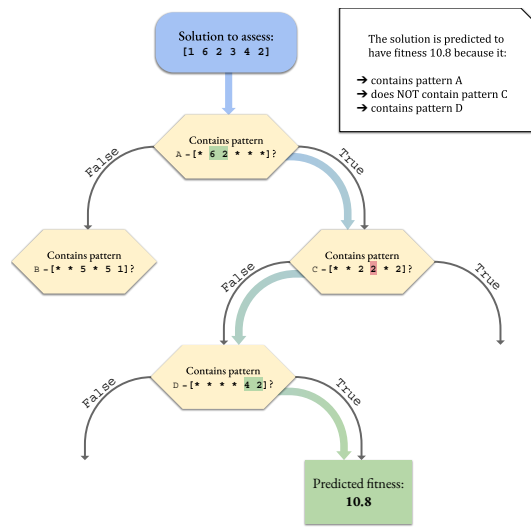


Figure 8.5: An example of a decision tree that is produced by our methodology, of depth 3. Here, a solution is passed to the model by checking which patterns it contains (which are the explanations), and a leaf node suggests the predicted fitness.

### 8.3.1 What is the issue with existing transparent surrogate models?

This research arises out of the unsuitability of existing transparent models. Certain kinds of models are deemed transparent and thus they are thought to be always understandable, which is not true [94, 159, 195], and it is discussed further in Section 3.7.3.

It is true that some models have a much better chance at being understandable, with **decision trees** being a common example. The simple rules used by these models are much more palatable than the format seen in more complex models [142]. **Decision trees** (discussed earlier in Section 3.5.1) strike a balance between generality (as opposed to linear models) and conciseness (as opposed to k-Nearest Neighbours), and thus they are picked as a transparent model which can be enhanced via **PSs**.

**ISSUES WITH EXISTING DECISION TREES** In the case of traditional decision trees, **excessive depth** is the main cause of lack of understandability, but it is often required in order to obtain a sufficiently accurate tree. It is well known in the literature that this depth is primarily caused by two limitations [23, 171]:

1. Every decision node can only make a decision based on **one variable**

2. Finding the best decision tree (with restricted depth) is an NP-complete task

The first issue is not a necessary evil, and there are many works which produce decision trees that make their decisions based on multiple variables at the same time [126, 285, 22]. Our focus here is not performance, but rather in understandability, and although the aforementioned multi-variable trees can be shallower thanks to more powerful decision nodes, the nodes themselves effectively become black boxes.

Other works instead focus on finding shallow trees which still use simple decision nodes, either by finding the globally optimal trees (with depth constraints) [162, 181, 23], or by finding deep trees and then removing the least-important splits from an existing tree (“pruning”) [224, 89]. These works focus on understandability, and the current state of the art is Interpretable AI’s implementation [22, 23].

PSs are the natural response to the challenge of understandable decision trees: checking if a pattern is present is human-understandable, and it is an assessment based on multiple variables at the same time. Moreover, we are interested in explaining solutions to optimisation problems rather than producing a general machine learning model, and PS are designed for this domain.

### 8.3.2 Methodology

The implementation of PS-based decision trees follows a recursive process, as shown in Algorithm 19, which mimics the traditional approach to decision tree building (implemented by ID3 and CD4.5 [225]).

Traditional greedy methods of inducing decision trees work by finding the criterion (based on one variable) that best partitions a set of points, and recursively splitting the resulting partitions. Our methodology will use the  $P_{\text{Ref}}$  from the metaheuristic method as the set of points, and will greedily find the splitting criteria (a PS) via a stochastic search method. Algorithm 19 can be summarised as follows:

1. Solve the optimisation problem using a metaheuristic, and obtain a  $P_{\text{Ref}}$ ;
2. Given the current  $P_{\text{Ref}}$ , find the PS which best partitions it. The pattern needs to satisfy the usual objectives of simplicity and atomicity, but it also requires a quantification of partition quality. As used in traditional decision trees for regression, **weighted variance** is used as the objective to be optimised. After running the PS mining process and selecting one PS (see Section 6.6.3), the decision node is determined.
3. The PS is then used to partition the current  $P_{\text{Ref}}$ . The corresponding  $P_{\text{match}}$  and  $P_{\text{notmatch}}$  are used to generate the right and left branch respectively, being considered the current  $P_{\text{Ref}}$  for those branches.
4. When the current  $P_{\text{Ref}}$  has few solutions within it (below a threshold) or a sufficient depth is reached, a leaf node is produced. This leaf node contains a fitness prediction, equal to the average fitness of the solutions in the current  $P_{\text{Ref}}$ .

**Weighted variance** (defined in Section 6.7 for PSs) is a metric that is already in use within many greedy decision tree heuristics, and is used to determine the quality of the splits produced. It acts as a replacement for the **mean-fitness** metric used in our previous work, since the intention here is to find strong predictors for fitness rather than for good fitness only.

Two variants of this methodology are considered, differing in objectives used.

- **PS-SWA**: \*-count simplicity, weighted variance and (Mutual Information of Survival) atomicity
- **PS-SW**: \*-count simplicity and weighted variance

This tree can be used to make predictions of fitness for solutions. This process is described in Algorithm 20 but it does not differ significantly from traditional decision trees for regression.

In the PS mining run used to build the tree, the evaluation budget is set to 5000, with a population size of 100, mutation rate of  $1/L$  and probability of crossover set to 70%. These parameters were informed by our research in [46].

---

**Algorithm 19** `get_tree(population, max_depth)`

---

```
1: function GET_TREE(population, max_depth)
2:   if max_depth = 0 or variance(fitness(population)) = 0 then
3:     return LeafNode(prediction = average(fitness(population)))
4:   end if
5:   best_solution ← max(population, by = fitness)
6:   split_ps ← search_ps(pattern_of = best_solution,
7:                       reference_population = population,
8:                       metrics = M,
9:                       settings = S)
10:  match_solutions ← [sol for sol in population if sol.contains(split_ps)]
11:  not_match_solutions ← set_difference(population, match_solutions)
12:  return Branching(split_pattern = split_ps,
13:                  matching = get_tree(match_solutions, max_depth-1),
14:                  not_matching = get_tree(not_match_solutions, max_depth-1))
15: end function
```

---

---

**Algorithm 20** `predict(current_branch, solution)`

---

```
1: function PREDICT(current_branch, solution)
2:   if isinstance(current_branch, LeafNode) then
3:     return current_branch.prediction
4:   else if solution.contains(current_branch.split_pattern) then
5:     return predict(current_branch.matching, solution)
6:   else
7:     return predict(current_branch.not_matching, solution)
8:   end if
9: end function
```

---

### 8.3.3 Experiments

The experiments were designed to determine how well our decision trees based on PSs can predict solution fitness, compared to other decision tree approaches. Additionally, the decision trees produced by our methods are analysed to check how complex the splitting PSs are.

For every optimisation problem and metaheuristic, we ran the search method 100 times and obtained 100 sets of solutions. For each of these sets, we split them into training and test solutions, and we train each of the trees (at different depths). Finally, the trees are evaluated by getting the  $R^2$  value on the test solutions, and by counting the average quantity of variables contained in the PSs. Note that all trees are trained and evaluated on the same 100 sets of solutions for each problem, to avoid unfair comparisons.

**BENCHMARKS** The experiments used **6 problem instances as benchmarks**: 3 SAT problems, 2 Graph Colouring and 1 Staff Rostering problem. These were selected in order to test our methods on problems of different scales and formulations. The SAT problems are from SATLIB [135], and the instances `uf-20-1`, `uf-50-1` and `uf-100-1` were picked to have 20, 50 and 100 variables respectively. (In the results, these are indicated as `SAT_20`, `SAT_50`, `SAT_100`) [135]. The Graph Colouring problems are two instances from Donald Knuth’s “book” graphs, one based on Tolstoy’s Anna Karenina (`GC_anna`, with 138 variables) and the other on Victor Hugo’s Les Misérables (`GC_jean`, with 80 variables) [270]. For each book, a connected graph is obtained by representing every character as a node, and with connections representing social or family connections between characters. Because these graphs are effectively social networks, they are a suitable simulacrum of what kinds of problems might be seen in real-world scenarios.

Finally, the Staff Rostering problem is the same as discussed in our previous work on PSs [46] and by trajectory-based research [96], with 141 variables.

**METAHEURISTICS** **4 different metaheuristics** are used to solve the optimisation problems and generate the training/test data for the trees. These will be Random Search (RS), a standard GA ( $\mu + \lambda$ ,  $\mu = \lambda = 100$ , uniform mutation, 70% chance of crossover), Simulated Annealing SA (as described by [81] for the Staff Rostering problem), and finally Tabu Search as defined in [185]. Each of these methods will be applied with a fixed evaluation budget of  $10^4$ , meaning that the resulting sets of solutions will be of size  $10^4$ . These are solution-fitness pairs which will be split into training (80%) and testing (20%) data (randomly).

**DECISION TREES** The decision trees that will be considered are:

- traditional trees by `scikit-learn` (based on the CART algorithm, [214])
- IAI (Interpretable AI) trees based on NIST's Optimal Regression trees [23].
- PS-SWA and PS-SW, as defined in Section 8.3.2.

The depths considered are 3, 4 and 5, since decision trees of greater depths are unlikely to be understandable [142].

#### 8.3.3.1 Summary of parametrisations

The parametrisations used for this evaluation are:

- benchmark problem  $\in$  [SAT\_20, SAT\_50, SAT\_100, GC\_anna, GC\_jean];
- metaheuristic applied on the benchmark problem  $\in$  [Random search, Simulated Annealing (with mutation), Tabu Search (population size = 100, tabu list length = 10, with mutation), Genetic Algorithm (population size = 100, with mutation and crossover, crossover rate = 0.5)] all with fitness evaluation budget of 10000, (where used) mutation operator is uniform mutation with rate of  $\frac{1}{L}$ , (where used) crossover is simulated binary crossover;
- maximum tree depth  $\in$  [3, 4, 5];
- combinations of objectives used for PS search  $\in$  [SW, SWA];
- other kinds of trees considered  $\in$  [CART (traditional), IAI]

#### 8.3.3.2 Measured outcomes

We measure both the accuracy of the trees and their complexity. For the accuracy, we collect the  $R^2$  value for the prediction vs ground truth fitnesses of the solutions in the test sets, and we perform statistical tests (Mann-Whitney U) to verify if PS-SW trees have significantly higher accuracy than traditional trees.

The **complexity** of the splitting criteria for PS-SW and PS-SWA is measured by counting the number of variables present in the pattern of the splitting nodes. We present the averages for each depth to investigate if they get more complex at different levels.

#### 8.3.4 Results

The key results seen in Table 8.4 and Table 8.5 are summarised as follows:

- PS-SW generally has more accurate predictions (higher  $R^2$ ) than PS-SWA and other methods;
- PS-SWA is comparable or slightly worse than traditional CART trees in performance, and its splitting criteria involve more variables than PS-SW;
- GA and SA result in decision trees with significantly improved predictive power for their own solutions.

Each of these points is expanded upon in the subsections below.

**PS-SW IS GENERALLY MORE ACCURATE** The results shown in Table 8.4 and Table 8.5 show the average  $R^2$  that results when comparing the ground truth fitness values against the predicted values. PS-SW is generally comparable to traditional decision trees, being either better or very close for GA and SA, and being less consistent for Tabu and Random Search. Statistical tests were carried out to verify whether our results were statistically significant, and the results in Table 8.6 suggest that PS-SW performs better for the GA in every scenario, and for SA as well in SAT problems.

The difference in  $R^2$  performance appears to be diminishing as the tree depth increases, and it is much higher for GA and SA than for Tabu and Random Search. This indicates that there are diminishing returns of using our methodology for deeper trees, although at different rates for different heuristics. For example, the average improvement observed for the GA is 9.27% at depth 2, 7.62% at depth 3, 6.07% at depth 4 and 4.38% at depth 5.

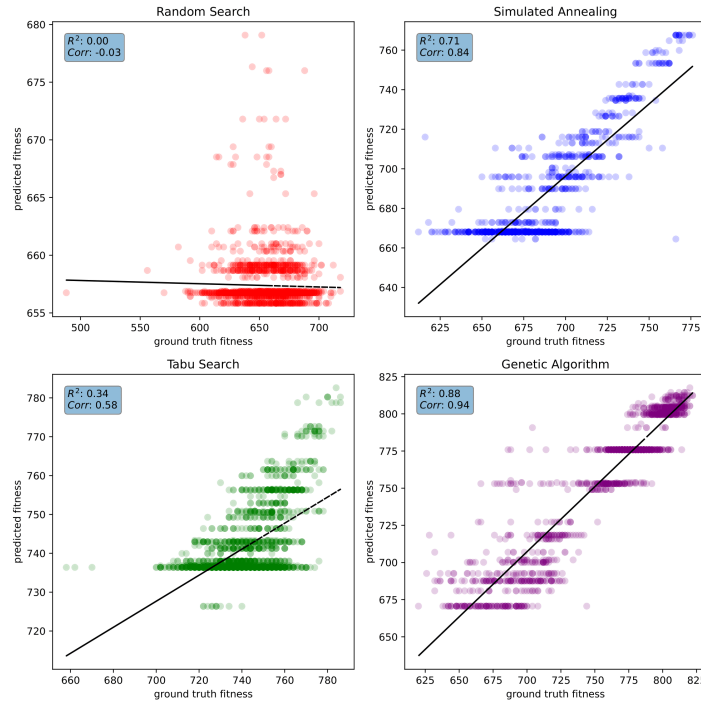


Figure 8.6: A comparison of how the search method affects the decision tree model. These scatterplots compare the true fitness of solutions against the predictions, along with a correlation line. Our decision trees strongly correlate with the ground truth when the search method is a GA, SA. The data shown is for GC\_anna, which was solved using 4 different search methods, and the evaluated solutions were split into training and test data (80%-20%).

**PS-SWA IS COMPARABLE TO TRADITIONAL METHODS** PS-SWA performs slightly worse but is generally on par with PS-SW. Looking at the complexity of the PSs in the splits, by checking the number of variables they involve as seen in Figure 8.7, it can be seen that PS-SWA produces more complicated patterns, which have a slight tendency to increase in size with depth. This is likely to be caused by the atomicity metric pushing for strongly linked subsets of variables that are not as important for prediction power.

**CERTAIN METAHEURISTICS RESULT IN BETTER PREDICTIONS** Another result is that the predictive power of the decision trees, for all methods, is **heavily dependent on the metaheuristic originally used**. GAs tends to yield the best  $R^2$ , followed by SA. There is then a large decrease in performance when using Tabu, and Random Search has widely varying results which are almost always the worst.

This effect is likely explained by the **convergence properties** of the different algorithms, with more strongly converging methods giving more “training data” to the decision trees to allow them to specialise in high fitness solutions. A method such as Random Search will not converge, and thus positive traits are under-represented. A visual representation of this effect is presented in Figure 8.6, where GA and SA produce more data-points in the high-fitness region.

This effect might also be a sign that an algorithm lacks diversity in its solutions, which also aligns with the procedure of Tabu search being designed to enhance diversity [185] and with Random Search being the least converging of the methods.

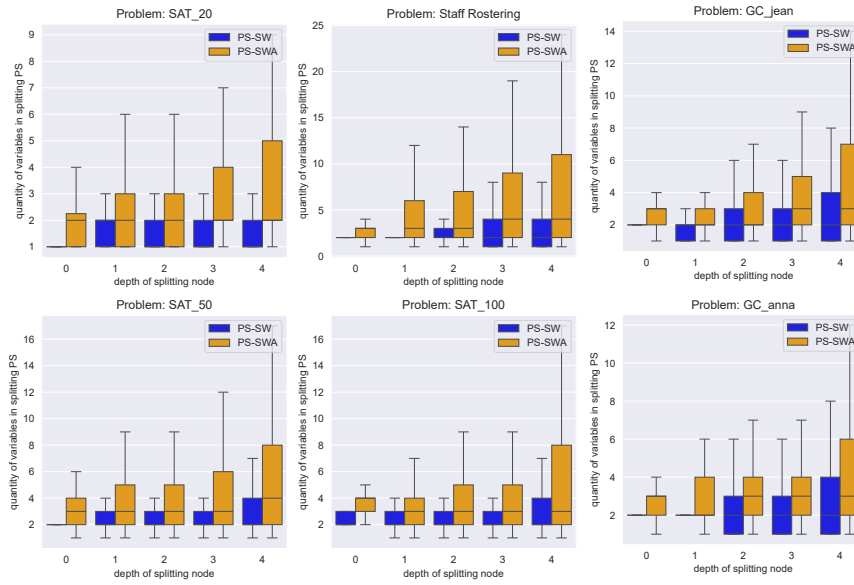


Figure 8.7: Box-plots showing the distribution of the “complexity” of the splitting criteria at different depths for PS-SW (blue) and PS-SWA (orange, on the right) for the benchmark problems (where depth 0 is the root node). These are for instances where the problem was solved using a GA, with 100 trials per combination.

Table 8.4: (average)  $R^2$  score for each decision tree, applied Staff Rostering, GC\_anna and GC\_jean. Values in bold are the highest per row (i.e. which decision tree performed best).

| problem         | Metah. | depth | PS-SW        | PS-SWA | IAI          | Trad.        |
|-----------------|--------|-------|--------------|--------|--------------|--------------|
| Staff Rostering | GA     | 3     | <b>85.1%</b> | 81.3%  | 66.6%        | 77.3%        |
|                 |        | 4     | <b>89.1%</b> | 86.2%  | 78.0%        | 83.4%        |
|                 |        | 5     | <b>91.2%</b> | 89.1%  | 83.8%        | 86.9%        |
|                 | SA     | 3     | 78.6%        | 72.0%  | 65.0%        | <b>79.2%</b> |
|                 |        | 4     | 84.2%        | 80.3%  | 82.1%        | <b>88.0%</b> |
|                 |        | 5     | 88.1%        | 86.1%  | 89.8%        | <b>92.8%</b> |
|                 | Tabu   | 3     | 26.0%        | 24.1%  | 21.9%        | <b>30.3%</b> |
|                 |        | 4     | 34.3%        | 32.7%  | 32.2%        | <b>39.3%</b> |
|                 |        | 5     | 42.5%        | 41.0%  | 41.8%        | <b>48.8%</b> |
|                 | RS     | 3     | 3.6%         | 5.2%   | 11.2%        | <b>14.5%</b> |
|                 |        | 4     | 4.9%         | 6.1%   | 14.3%        | <b>17.2%</b> |
|                 |        | 5     | 6.1%         | 6.7%   | 16.7%        | <b>18.9%</b> |
| GC_anna         | GA     | 3     | <b>75.7%</b> | 72.9%  | 56.9%        | 67.1%        |
|                 |        | 4     | <b>80.3%</b> | 77.9%  | 67.9%        | 73.3%        |
|                 |        | 5     | <b>83.0%</b> | 81.4%  | 74.6%        | 77.4%        |
|                 | SA     | 3     | <b>47.3%</b> | 42.1%  | 31.4%        | 45.2%        |
|                 |        | 4     | 55.9%        | 51.2%  | 49.3%        | <b>59.2%</b> |
|                 |        | 5     | 62.9%        | 59.3%  | 63.0%        | <b>70.8%</b> |
|                 | Tabu   | 3     | <b>11.2%</b> | 9.6%   | 6.4%         | 10.5%        |
|                 |        | 4     | 17.1%        | 14.2%  | 12.9%        | <b>17.7%</b> |
|                 |        | 5     | 23.7%        | 19.6%  | 20.7%        | <b>27.7%</b> |
|                 | RS     | 3     | -0.5%        | -0.5%  | -0.1%        | <b>-0.0%</b> |
|                 |        | 4     | -1.1%        | -0.8%  | <b>-0.0%</b> | -0.2%        |
|                 |        | 5     | -2.0%        | -1.4%  | <b>0.2%</b>  | -1.0%        |
| GC_jean         | GA     | 3     | <b>82.6%</b> | 80.3%  | 63.3%        | 74.2%        |
|                 |        | 4     | <b>86.5%</b> | 84.5%  | 75.0%        | 80.4%        |
|                 |        | 5     | <b>88.8%</b> | 87.4%  | 81.5%        | 84.3%        |
|                 | SA     | 3     | 65.2%        | 58.7%  | 52.0%        | <b>67.5%</b> |
|                 |        | 4     | 73.3%        | 68.2%  | 71.9%        | <b>79.3%</b> |
|                 |        | 5     | 79.4%        | 75.4%  | 82.2%        | <b>86.4%</b> |
|                 | Tabu   | 3     | 22.6%        | 19.5%  | 16.1%        | <b>24.1%</b> |
|                 |        | 4     | 31.5%        | 27.5%  | 28.3%        | <b>36.6%</b> |
|                 |        | 5     | 40.7%        | 35.7%  | 40.9%        | <b>49.8%</b> |
|                 | RS     | 3     | -0.9%        | -0.7%  | <b>-0.1%</b> | -0.4%        |
|                 |        | 4     | -1.5%        | -1.2%  | <b>-0.1%</b> | -1.0%        |
|                 |        | 5     | -2.6%        | -1.9%  | <b>-0.1%</b> | -2.4%        |

Table 8.5: (average)  $R^2$  score for each decision tree, applied to SAT\_100, SAT\_50 and SAT\_20. Values in bold are the highest per row (i.e. which decision tree performed best).

| problem | Metah. | depth | PS-SW        | PS-SWA | IAI   | Trad.        |
|---------|--------|-------|--------------|--------|-------|--------------|
| SAT_100 | GA     | 3     | <b>77.8%</b> | 75.5%  | 50.0% | 62.1%        |
|         |        | 4     | <b>83.3%</b> | 80.6%  | 62.2% | 70.2%        |
|         |        | 5     | <b>86.4%</b> | 83.9%  | 70.5% | 75.7%        |
|         | SA     | 3     | <b>65.1%</b> | 56.1%  | 36.6% | 52.1%        |
|         |        | 4     | <b>73.9%</b> | 65.5%  | 56.2% | 67.0%        |
|         |        | 5     | <b>80.4%</b> | 73.6%  | 70.9% | 79.4%        |
|         | Tabu   | 3     | <b>21.8%</b> | 16.6%  | 10.2% | 15.8%        |
|         |        | 4     | <b>31.0%</b> | 23.8%  | 17.2% | 23.6%        |
|         |        | 5     | <b>40.5%</b> | 31.9%  | 25.1% | 33.4%        |
|         | RS     | 3     | 6.8%         | 3.6%   | 5.7%  | <b>7.6%</b>  |
|         |        | 4     | 8.2%         | 4.7%   | 7.4%  | <b>9.1%</b>  |
|         |        | 5     | 8.9%         | 5.0%   | 9.0%  | <b>10.2%</b> |
| SAT_50  | GA     | 3     | <b>68.2%</b> | 63.9%  | 40.3% | 51.7%        |
|         |        | 4     | <b>74.8%</b> | 70.8%  | 52.3% | 59.5%        |
|         |        | 5     | <b>78.8%</b> | 75.2%  | 60.3% | 65.6%        |
|         | SA     | 3     | <b>50.5%</b> | 42.6%  | 27.1% | 38.7%        |
|         |        | 4     | <b>59.3%</b> | 51.5%  | 40.4% | 50.2%        |
|         |        | 5     | <b>66.9%</b> | 59.5%  | 52.9% | 61.6%        |
|         | Tabu   | 3     | <b>18.4%</b> | 13.3%  | 9.8%  | 14.0%        |
|         |        | 4     | <b>24.3%</b> | 18.6%  | 14.6% | 18.9%        |
|         |        | 5     | <b>30.8%</b> | 24.6%  | 19.6% | 24.6%        |
|         | RS     | 3     | 12.6%        | 7.9%   | 9.2%  | <b>14.1%</b> |
|         |        | 4     | 15.6%        | 10.3%  | 14.5% | <b>18.2%</b> |
|         |        | 5     | 17.9%        | 12.1%  | 18.4% | <b>21.5%</b> |
| SAT_20  | GA     | 3     | <b>41.1%</b> | 29.9%  | 31.8% | 40.2%        |
|         |        | 4     | <b>47.7%</b> | 37.7%  | 40.2% | 47.2%        |
|         |        | 5     | 52.0%        | 45.1%  | 48.2% | <b>52.4%</b> |
|         | SA     | 3     | 29.1%        | 23.5%  | 27.3% | <b>36.1%</b> |
|         |        | 4     | 35.9%        | 31.1%  | 37.6% | <b>45.2%</b> |
|         |        | 5     | 41.3%        | 38.8%  | 47.3% | <b>53.2%</b> |
|         | Tabu   | 3     | 20.1%        | 14.3%  | 15.5% | <b>20.4%</b> |
|         |        | 4     | 24.9%        | 21.3%  | 21.6% | <b>26.2%</b> |
|         |        | 5     | 29.9%        | 28.0%  | 27.8% | <b>31.8%</b> |
|         | RS     | 3     | 14.0%        | 17.5%  | 24.2% | <b>33.4%</b> |
|         |        | 4     | 15.0%        | 23.4%  | 33.5% | <b>41.2%</b> |
|         |        | 5     | 15.7%        | 29.3%  | 41.8% | <b>47.3%</b> |

Table 8.6: The performance of PS-SW trees and traditional trees is compared. For every combination (problem, depth, metaheuristic), we report the difference in average  $R^2$  values and the p-value for the test of differences of averages (Mann-Whitney-U, single-tailed). Cells in bold are those which have a p-value below 0.005.

| Prob.    | depth | 3                    | 4                    | 5                    |
|----------|-------|----------------------|----------------------|----------------------|
|          | Met.  |                      |                      |                      |
| Staff R. | GA    | <b>7.9%, 0.001</b>   | <b>5.85%, 0.001</b>  | <b>4.45%, 0.001</b>  |
|          | SA    | -0.5%, 0.438         | -3.72%, 1.0          | -4.7%, 1.0           |
|          | Tabu  | -4.01%, 0.994        | -4.69%, 0.989        | -6.01%, 0.998        |
|          | RS    | -10.91%, 1.0         | -12.34%, 1.0         | -12.86%, 1.0         |
| GC_anna  | GA    | <b>8.43%, 0.001</b>  | <b>6.83%, 0.001</b>  | <b>5.5%, 0.001</b>   |
|          | SA    | <b>2.03%, 0.012</b>  | -3.48%, 1.0          | -7.91%, 1.0          |
|          | Tabu  | <b>0.83%, 0.013</b>  | -0.54%, 0.898        | -3.9%, 1.0           |
|          | RS    | -0.53%, 1.0          | -0.99%, 1.0          | -1.1%, 1.0           |
| GC_jean  | GA    | <b>8.34%, 0.001</b>  | <b>6.07%, 0.001</b>  | <b>4.49%, 0.001</b>  |
|          | SA    | -2.44%, 0.964        | -6.06%, 1.0          | -7.07%, 1.0          |
|          | Tabu  | -1.56%, 0.991        | -5.06%, 1.0          | -8.98%, 1.0          |
|          | RS    | -0.45%, 1.0          | -0.55%, 1.0          | -0.18%, 0.819        |
| SAT_100  | GA    | <b>15.92%, 0.001</b> | <b>13.11%, 0.001</b> | <b>10.79%, 0.001</b> |
|          | SA    | <b>13.22%, 0.001</b> | <b>7.02%, 0.001</b>  | <b>1.16%, 0.003</b>  |
|          | Tabu  | <b>5.94%, 0.001</b>  | <b>7.27%, 0.001</b>  | <b>7.04%, 0.001</b>  |
|          | RS    | -0.8%, 0.996         | -0.87%, 0.998        | -1.31%, 1.0          |
| SAT_50   | GA    | <b>16.69%, 0.001</b> | <b>15.37%, 0.001</b> | <b>13.32%, 0.001</b> |
|          | SA    | <b>11.74%, 0.001</b> | <b>9.04%, 0.001</b>  | <b>5.26%, 0.001</b>  |
|          | Tabu  | <b>4.4%, 0.001</b>   | <b>5.31%, 0.001</b>  | <b>6.21%, 0.001</b>  |
|          | RS    | -1.57%, 1.0          | -2.56%, 1.0          | -3.66%, 1.0          |
| SAT_20   | GA    | <b>1.11%, 0.001</b>  | <b>0.52%, 0.003</b>  | <b>-0.34%, 0.017</b> |
|          | SA    | -7.01%, 1.0          | -9.23%, 1.0          | -11.97%, 1.0         |
|          | Tabu  | <b>-0.41%, 0.001</b> | <b>-1.4%, 0.014</b>  | <b>-1.97%, 0.007</b> |
|          | RS    | -19.28%, 1.0         | -26.19%, 1.0         | -31.55%, 1.0         |

To conclude this section, the main aspects of Chapter 3 are revisited here in order to determine what aspects of explainability are relevant to this research.

#### 8.4.1 *What is being explained?*

The patterns explain the *solutions* produced by the metaheuristic, rather than the metaheuristic itself. Making the algorithm understandable is a task that is beyond the scope of this research. The explanations can relate to one solution at a time (local explanations) or to a set of solutions which are a representative sample of the search space (global explanations). The explanation shows that the presence of certain patterns is the *likely cause* of high fitness.

#### 8.4.2 *Are the patterns understandable?*

PSs are simple and transparent in the same way that decision trees are: their format lends itself well to human rules, but their size needs to remain below a certain threshold.

The mechanism which generates patterns is a **search method that uses “simplicity” as one of its objectives**, therefore the algorithm seeks to find the simplest patterns it can. Simplicity in this form is the main driver for understandability, but another definition for simplicity is provided in Section 6.8: **independence**.

These are proxy metrics for understandability, and the user study proposed in [42] (to be discussed in the next chapter) was designed to verify this more concretely, but unfortunately the results could not be used.

Although the results of the study are not usable, it can still be argued that the patterns are understandable in most cases, because their complexity is tunable. This happens in two ways:

- There are many objectives to choose from, and the user can pick the ones that they understand the most, or implement custom metrics. Simplicity is not the only objective relevant here, as certain metrics of fitness quality might also be more “understandable” than others.
- The results produced by our system are a Pareto front of patterns. Provided that the PS mining method was given enough evaluation budget, within the Pareto front there will always be a pattern that is *simple enough* (if simplicity is one of the objectives).

The short answer is then “whatever threshold of complexity the user holds, the system will be able to produce a pattern below that complexity”.

Additionally, the process of understanding patterns is aided by the presence of **descriptors**. This feature aligns with the idea of reducing the effort required to understand information.

#### 8.4.3 *Are the explanations true?*

A pattern is taken to be “the reason for high-fitness in solutions that contain it”, and thus it can be considered true if this relationship is proven. The patterns found by our system can be shown to be associated with improved fitness in solutions, using statistical testing. If a user wants to be sure that the explanations are true, the **mean-fitness** metric can be replaced by the metrics which are p-values to statistical tests. By doing that, the system explicitly searches for statistically provable explanations.

The patterns found in PS4XRS were shown to be able to “generalise” in the ML sense, meaning that they perform as well on unseen data as on training data, [43].

A model built on PSs can only have satisfactory predictive accuracy if the PSs are “true”, and the experimentation in Section 8.3 shows for this to occur.

If one builds a predictive model of fitness using patterns, the predictive performance is high enough to suggest that the patterns are capturing correct information [45]. Therefore, there is evidence suggesting that the PSs found by our system are capturing true information relating to solutions.

#### 8.4.4 *Do Partial Solutions have practical advantages?*

Knowledge of significant **PSs** can inform various actions. Firstly, a solution’s fitness may be improved by applying a beneficial pattern (and this can be ensured to work if the pattern satisfies certain objectives). In other cases the pattern is already present, and the user will now know that it should not be altered in case of changes being required. If there is a “loophole” in the fitness function, our system will easily find the minimum requirements to satisfy it, thus informing its resolution. Rather than using anecdotal evidence, a user study would have been more suitable to prove the actionability and utility of our methodology. This was designed in [42] but it was not possible to carry it out to completion. A full definition and discussion of this user study is presented in Chapter 9.

#### 8.4.5 *Is the process interactive and customisable? Is there trial and error?*

Interactivity of our system is intended to mean the user has sufficient control to obtain results that align with their needs. The system is highly modular, and therefore there are many settings which the user can customise (what dataset, what search objectives, which pattern from the results). For this to work in practice, the system needs to be sufficiently fast, and thus a lot of care went into avoiding bottlenecks when finding **PSs**, and it was evaluated in Section 6.14.

The user is able to try many combinations of search metrics and datasets, before finding the “protocol” of explanations that best suits them.

#### 8.4.6 *Is there a ground truth? Is it known if this system finds the ground truth?*

There are many ways to define ground truth in this context. The most obvious case is that there should be certain optimal patterns and that we are finding them, which was shown to be achieved in [46] (Section 6.6.1 in this thesis).

A more detailed overview is in Section 6.11, where it was shown that “ground truth patterns” are in many cases a subjective choice but can still be found by using the right objectives.

Such patterns are easily achieved by our system (as discussed in relation to whether the patterns are true).

In general, it is hard to tell what would make a **PS** perfect, which is an issue inherited from explainability as a whole. In fact, it is impossible to tell if a given explanation is the best possible explanation for a certain context [276]. Similarly to how this is resolved in optimisation, the question of “Is the correct explanation being found?” should instead be “Is the obtained explanation good enough?”. The answer to this last question is more nuanced, but the answers were provided in this chapter.

This chapter presents a contribution published in [42], in which we propose a methodology for evaluating PSs for explanations. Explanations are designed for human use, and although they can be evaluated using proxy metrics, evaluation must determine whether their presence is beneficial.

As part of our research, we endeavoured to create a **user study** which tackles these challenges (mainly lack of concrete evidence) with particular attention to how they would relate to XAI for metaheuristics, in order to evaluate the utility of our explanations.

Our main contributions here include:

- a discussion on the evaluation of explanations for metaheuristics,
- the design process of a small-scale user study to evaluate metaheuristic explanations,
- a preliminary analysis of the results of the user study, evaluating PSs.

The user study discussed here evaluated the explanations introduced in [45] (decision trees with PS nodes). For this user study, the optimisation problem being used is Staff Rostering, and the post-processing of the resulting explanations use problem-specific information through the use of descriptors. An example of these explanations is shown in Figure 9.1).

## 9.1 MOTIVATIONS

There are many parallels between ML and metaheuristics (both are stochastic processes, they optimise a fitness function by changing parameters) but despite these similarities there is a marked disparity in their representation in XAI literature [300]. Review papers relating to XAI tend to **assume the presence of a model** and often define explanations in terms of models (such as [221, 183, 94, 276, 207, 196, 134, 12, 7, 233]), although XAI also applies to AI planning, Human-in-the-Loop methods, explainable agents and metaheuristics [3]. These assumptions make the application of the literature less transferable to metaheuristics, and the present document will touch on some of these incompatibilities.

The **evaluation** of explanations proves to be particularly challenging due to the subjectivity of perceived explanation quality and in designing methods which allow comparisons between different kinds of explanations [252, 183, 207]. The literature is not easily applicable to XAI for metaheuristics, and to the best knowledge of the author there is no existing study on the evaluation of explanations for metaheuristics involving human participants.

The **user study** presented here is therefore novel in that it is an evaluation of explanations for a metaheuristic. The main novelty of this study is its focus on practical and provable usefulness - aiming to answer the question “**Are our explanations quantifiably useful for human users?**”.

## 9.2 OUR SPECIFIC SCENARIO

We focus on a specific task, informed by our industrial partner. A search method produces solutions that need to be **manually modified** to fit unforeseen constraints, but the user is not sure of what modifications are least disruptive to fitness. These manual modifications allow for what has been called **malleability** in the literature [14]. More concretely, the solutions resulting from these manual modifications should have high fitness, higher than what would have been achieved without explanations.

XAI has been discussed already in Chapter 3, covering aspects such as evaluation and metaheuristics. Section 3.8 provided an overview of user studies and more generally evaluation of XAI.

We are interested in **local explanations** of solutions, in order to adapt them to last-minute requirements, which is of interest to our industrial partner.

Our industrial partner reports the fact that users are known to make counterproductive choices due to a lack of understanding and to a false sense of confidence, which motivates the need for explanations that are both trustworthy and straightforwardly applicable.

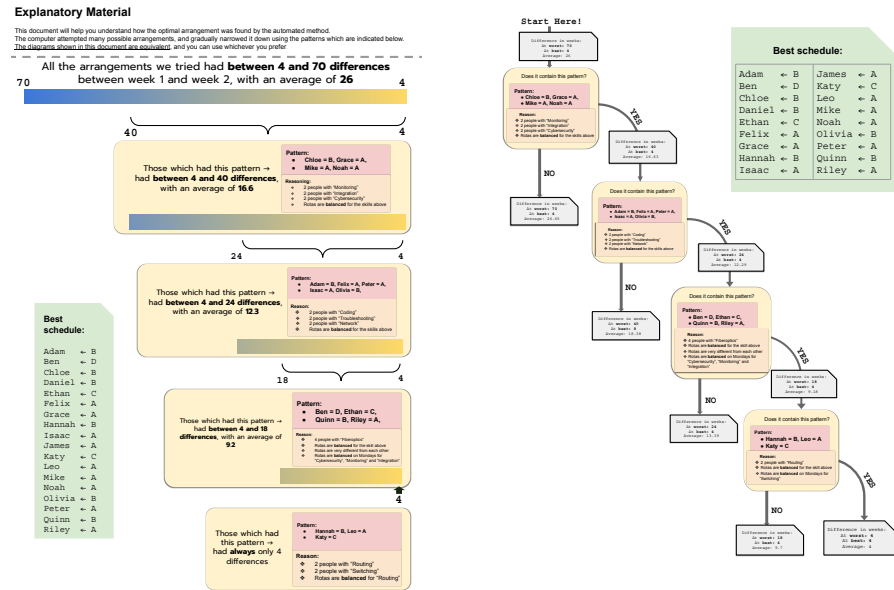


Figure 9.1: An example of an explanation provided to the user. The explanation consists of a shallow decision tree based on Partial Solutions (right). The tree structure might also be interpreted as a narrowing of the range of possible fitnesses of the solution (left), so both formats are presented.

### 9.2.1 Resource constraints

The resources for this user study were constrained in multiple ways:

- the **funding** available to reward participants was limited to 25 1-hour questionnaire sessions, or a handful of 1-to-1 interviews. In order to obtain sufficient samples, the questionnaire was chosen.
- the study needed to take **1 hour** or less per participant, which was a significant restriction - each user needs to give consent, read the overall instructions, and execute each experimental task. This limited the total quantity of tasks to be administered to each user to two.
- if 25 users were available at most, dividing the participants into more than 2 groups would have prevented statistical analysis. Due to this small sample size, it was necessary to gather as much data as possible from each individual student, which clashes with the 1-hour constraint.
- the Computer Science department at the University of Stirling has around 200 undergraduate students, and 200 postgraduate students. Removing all the students that are unlikely to participate, the total quantity is already severely limited. Therefore, limiting this user study to this department is not expected to produce sufficient samples, and it was instead advertised to the entire university. The final sample of participants is composed of roughly 50% Computer Science students, which will be taken into consideration during analysis.
- in order for the study to be advertised and delivered university-wide, it needed to be in the form of a questionnaire which participants could do at home. This prevented any interactive sessions, code execution, interviews and other kinds of interactions. Instead, it consisted of a set of documents with instructions and explanations, which were necessary in order to complete the questionnaire.

Understanding these restrictions is necessary in order to understand many choices taken in the design and deployment of this evaluation.

The process of designing the study was heavily influenced by [62] and [184], and involved 8 major steps:

1. Finding the use case scenarios to be used as reference
2. Planning of study metrics and audience
3. Designing the experimental procedure
4. Simplifying the optimization problem to suit the audience
5. Applying metaheuristics to the problem, and producing experimental materials
6. Designing the questions and tasks in the questionnaire
7. **Generating the explanations**
8. Deploying the user study

This section will describe each of these stages, with motivations, and their specific answer for our user study.

#### 9.3.1 *Use case scenarios*

One of the main criticisms put forward by [255] is that **XAI** evaluation is not representative of the real scenarios where explanations are intended to be used. Therefore, it is important to determine where and when the explanations are going to be used, and to design a user study to be as close as possible to these contexts.

In our case, the explanations are developed for our industrial partner, who needs to solve the Staff Rostering problem (defined in Section 2.2.7.1). Solutions solved by a metaheuristic (iRoster, defined in [81]) **often need to be manually modified to fit last-minute constraints**. These modifications will inevitably worsen the fitness, but explanations can help in minimising the difference.

#### 9.3.2 *Planning of study metrics and audience*

This section is aligned with the planning stage suggested in [62], which requires the researcher to decide the following:

- whether the study is **qualitative or quantitative**,
- the **goals** the explanations are intended for,
- the **target audience** for the explanations.

In our case, we focus on quantitative information in order to provide more concrete evidence of usefulness, but still collect some qualitative information in order to determine areas for improvement.

Following discussions with our industrial supervisor, our explanations are motivated by the need for **trust**, **satisfaction** (in interacting with the system) and **effectiveness** (in terms of the fitness caused by the user).

Finally, we decided the target audience to be students at the University of Stirling, since the availability of employees with our industrial partner was limited, although one engineer did participate in order to provide a baseline for completion time. The use of lay-users was therefore necessary, but it required simplifying the original optimisation problem. This framework follows a **human-grounded** study [83].

#### 9.3.3 *Design of Experimental procedure*

Here, the tasks given to the participant within the experiment are decided based on the previous stages, focusing on determining if the explanations allow **prediction** and **actionability**. For some participants, the tasks are assisted by explanations, whereas for others (the control group) there is no such support tool. The tasks have **measurable outcomes**, so that the two groups can be compared.

There is considerable risk for **bias** in the design of the tasks: the task might be accidentally designed to be only achievable if an explanation is provided. The tasks must be designed based on the scenarios found in the step, and *independently* from the explanations themselves.

#### 9.3.3.1 *The tasks*

Based on these insights, the tasks used in our study are based on modifying the optima produced by a metaheuristic. Two different tasks were devised:

- **Task 1: Ranking of modifications:** of 4 possible changes, the user needs to rank them from best to worst in terms of fitness. The user is not told the resulting fitness for each of these changes. The user performance is based on how well the user’s ordering matched the fitnesses’ real ordering
- **Task 2: Change of rotas:** the user has to change 2 employees to their preferred working patterns to minimise disruption, and they are given 5 employees to choose from. The user is not given the fitness resulting from each change. User performance is based on the fitness of the solution produced by their modification

As well as quantitative information, we also collect qualitative data by having the users select how much they agree (on a Likert scale) with certain statements, such as “I found the explanations to be useful”.

#### 9.3.4 *Simplification of the original optimisation problem*

As specified by [83], a human-grounded approach can make use of a **simplified task** in order to adapt it to a less-experienced target audience and to make the experiments take less time.

In our case, the original problem has a complex fitness function dependent on many tables with hundreds of rows of values, and thus many simplifications were needed. The definition of the problem is presented in Section 2.2.7.2, with parameters presented in Chapter E, and was the result of a long iterative process in order to produce a problem that was both hard enough to justify the need for metaheuristics but also easy enough that the lay-user could feasibly understand.

#### 9.3.5 *Solving the problem and generation of the user materials*

After designing the framework of the user study, it is then necessary to generate the specific tasks which the user will work through. It is important for this step to **not include the explanations yet**, in order to remove any chance of bias: the experimental tasks need to be independently designed from the explanations. For our study, the optimisation tasks were solved using our industrial collaborator’s custom metaheuristics (iRoster, with uniform mutation with rate =  $\frac{1}{L}$ , evaluation budget of 10000). The problem instance settings and optima were then to be presented to the participants in documents which were written in plain English without making assumptions of participant knowledge. Following the suggestion in [62], users were also “primed” with a **mental model** which can assist them in understanding the problem. This consisted of a graph structure representing each of the workers in the problem, being connected if they share a skill. This structure was chosen due to the already-known connection between Staff Rostering problems and Graph Colouring problems [253], and for a lay-user it makes it easier to understand which workers are connected when affecting the fitness value. This section, called “Tips and Tricks” is shown in Figure 9.2.

#### 9.3.6 *Design of questions*

Three main kinds of questions are to be asked in a user study of this kind:

- Demographic information, background and preferences
- Qualitative information: choices picked by the user for the tasks
- Quantitative information: opinions of the user for each task

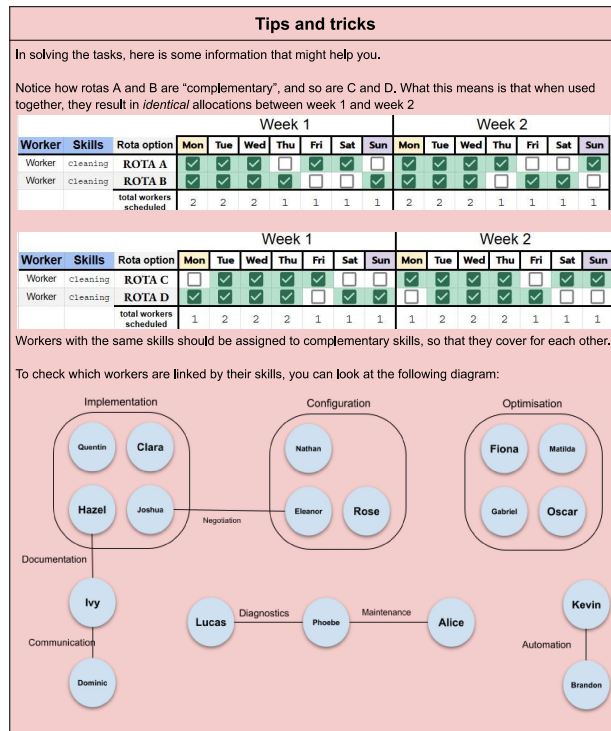


Figure 9.2: The “tips and tricks” section of the material available to the user, used to provide them a mental model of how they might tackle the optimisation problem

The first kind included questions to determine the user’s self-assessed numeracy skills, opinions on automated methods, and familiarity with optimisation.

The second kind of questions, relating to the experimental tasks, were created by selecting workers at random and generating questions based on them. The final kind, the subjective questions, included the following:

- “Did you **understand the task**?”
- “Did you **trust the metaheuristic** to solve the problem?”
- How did you feel about their answer?”
- (If the explanations were given) “Did you find the **explanations helpful/easy/accurate/trustworthy**?”

Additionally, the user had to provide a text explanation to the workers affected by their modification, which can be used to see their reasoning and to see if this was influenced by explanations.

Special care was put into writing the questions in such a way as to **avoid computer science terminology**. The real use cases relate to managers who are not familiar with metaheuristics, and they have no reason to think in terms of “solutions” or “fitness functions”. The questions and explanations used more concrete terms, such as “workers”, “availability”, “rotas” etc. The questionnaire is available in Chapter F.

### 9.3.7 Generation of explanations

The explanations are generated at this stage in order not to influence previous steps. In our case, explanations consist of the **decision trees** discussed in Section 8.3, which can be seen in Figure 9.1. The trees can also be represented as a progressive narrowing of fitness expectation, which is also shown to the user. The trees were generated with a maximum depth of 4, where each node was produced by a **PS** mining process with an evaluation budget of 10000. The objectives used are **mean-fitness**, **\*-count** **simplicity** and **atomicity** (using Mutual Information of Survival).

### 9.3.8 Implementation of experiment

The study was implemented by having **two questionnaires**, each having tasks 1 and 2, but differing in which tasks get an explanation. Questionnaire “A” had explanations for Task 1 but not Task 2, whereas Questionnaire “B” had explanations for Task 2 but not for Task 1. When users received the questionnaire, they were *randomly* assigned to one of the questionnaires.

A potential risk arises by having two tasks is that the user who received an explanation in their first task might learn to perform well in their second task *without an explanation*. To counter this, the problem instances between Task 1 and Task 2 were different (*i.e.* different workers and skill-sets) of equivalent complexity: the problems had exactly the same structure, but the variables were shuffled.

A critical issue was present in the design of the user study, which prevented its results from being usable: there is no mechanism to prevent participants from giving random answers in order to obtain the reward at the end. This issue should have been considered at this stage, and it will be discussed in Section 9.5.

### 9.3.9 Deployment of user study

Once the questionnaire(s) were completed, with use cases and explanations, they were then deployed to the users. The study was advertised to undergraduate students at the University of Stirling for a month, and then closed early due to noticing that many users were giving random answers and invalidating the study.

## 9.4 OUTCOME OF STUDY AND ANALYSIS OF RESULTS

The questionnaire was completed by 14 participants. Due to the monetary reward of our study, we inadvertently attracted some users who submitted **unreliable** answers, which meant that the study had to be interrupted. We decided to discard the data from any user who completed the entire study in less than 20 minutes, which brought the count of participants to 8. This threshold was chosen since in a trial run with a **BT** manager, it took 20 minutes. This filtering has caused the groups to be highly unbalanced (7 participants did questionnaire A vs 1 participant did questionnaire B), and therefore statistical testing cannot take place.

While this sample size is insufficient for statistical testing, it is still useful to present the statistical methods which would be used to analyse the quantitative results, so the *hypothetical* statistical testing is discussed.

### 9.4.1 Analysis of subjective questions

This section qualitatively analyses the participants’ subjective responses. The analysis is divided into a **quantitative evaluation of Likert-scale answers** and a **qualitative analysis of long-form text**. Although the sample size is small ( $N = 8$ , including only participants who completed the task appropriately), these insights provide a valuable human-centric evaluation of the methods presented in this thesis.

#### 9.4.1.1 Self-evaluation of skills and abilities

Participant self-assessments are summarized in Figure 9.3. Participants rated their abilities across three categories (“Basic”, “Limited”, “Intermediate”, “Proficient”, or “Expert”) and indicated their agreement with various statements (from “Strongly Disagree” to “Strongly Agree”).

Participants were asked, “How would you rate yourself in the following technical skills?”, based on three categories:

- **Numeracy:** working with numbers, basic arithmetic, and mathematical problems;
- **Problem Solving:** tackling technical challenges and applying creative solutions;
- **Algorithmic Thinking:** ability to break down complex problems and design step-by-step solutions.

Regarding numeracy, five participants selected “Proficient,” while “Limited,” “Intermediate,” and “Expert” were selected by one participant each. For problem-solving,

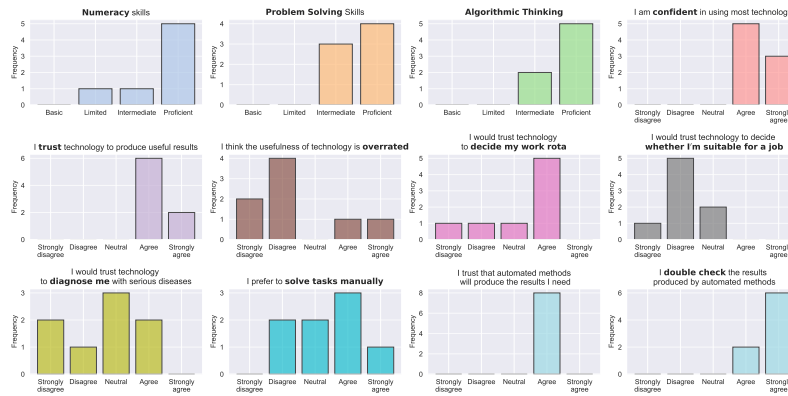


Figure 9.3: Histograms of questionnaire responses. Key insights indicate that participants are confident in their technical skills and generally accept automated methods for administrative tasks (e.g., work rotas), but remain skeptical of their use in critical applications.

responses varied: four participants selected “Proficient,” three “Intermediate,” and one “Expert.” Algorithmic thinking assessments similarly indicated high confidence: five selected “Proficient,” two “Intermediate,” and one “Expert.” None of the participants selected “Basic” for any category.

These self-assessments suggest the cohort was comfortable with the foundational skills required for the study. However, two factors should be noted: **self-selection bias** and the **Dunning-Kruger effect** [237]. Computing students self-select into their degree and may overestimate their expertise relative to the general population or industry professionals. While this potential bias does not invalidate the study, it provides necessary context for evaluating explainability metrics.

#### 9.4.1.2 *Sentiment towards technology and automated methods*

Participants answered questions regarding the practical application of technology and automated methods:

- “I trust technology to produce useful results”: 6 Agree, 2 Strongly Agree.
- “I think the usefulness of technology is overrated”: 2 Strongly Disagree, 4 Disagree, 1 Agree, 1 Strongly Agree.
- “I trust that automated methods will produce the results I need”: 8 Agree.

Participants generally agreed that technology is useful. Furthermore, they indicated strong confidence in the “actionability” of automated methods to achieve desired outcomes.

A subsequent set of questions examined trust in automated decision-making:

- “I would trust technology to decide my work rota”: 1 Strongly Disagree, 1 Disagree, 1 Neutral, 5 Agree.
- “I would trust technology to decide whether I’m suitable for a job”: 1 Strongly Disagree, 5 Disagree, 2 Neutral.
- “I would trust technology to diagnose me with serious diseases”: 2 Strongly Disagree, 1 Disagree, 3 Neutral, 2 Agree.

Trust varied by application context. Participants were willing to accept algorithmic decision-making for administrative tasks like scheduling, but remained averse to its use for critical decisions such as hiring or medical diagnoses.

A final set of questions assessed attitudes toward manual versus automated workflows:

- “I am confident in using most technology”: 5 Agree, 3 Strongly Agree.

- “I prefer to solve tasks manually”: 2 Disagree, 2 Neutral, 3 Agree, 1 Strongly Agree.
- “I double check the results produced by automated methods”: 2 Agree, 6 Strongly Agree.
- “I prefer to solve tasks by myself rather than using a computer”: 2 Disagree, 2 Neutral, 3 Agree, 1 Strongly Agree.

While participants reported high confidence in using technology, they demonstrated a persistent desire for human oversight. Every participant indicated they double-check automated results, and half expressed a preference for manual problem-solving, underscoring a “trust but verify” approach.

#### 9.4.1.3 *Opinions on provided materials*

Participants were presented with materials explaining the Staff Rostering problem, the optimized solution, the automated method, and the explanations. Four questions assessed comprehension and trust:

- “I understand the general structure of the problem”: 1 Disagree, 1 Neutral, 5 Agree, 1 Strongly Agree.
- “I trust that the solution produced by the computer is good”: 5 Agree, 3 Strongly Agree.
- “I believe that I could make a better rota assignment than the computer”: 3 Disagree, 3 Neutral, 2 Strongly Agree.
- “I understand the additional explanation material”: 7 Agree, 1 Neutral.

Participants indicated clear comprehension of both the problem structure and the provided explanations, and they expressed direct trust in the algorithmic solution.

#### 9.4.1.4 *Analysis of Task 1 (with explanations)*

After completing the first task using the provided explanations, participants wrote a textual justification aimed at the affected workers. Qualitative content analysis was conducted on these responses [92], generating the codebook in Table 9.1.

| Theme                   | Example Quote                                               | Freq. ( $N = 8$ ) |
|-------------------------|-------------------------------------------------------------|-------------------|
| Skill similarity        | “Since Isaac and Olivia share skills. . .”                  | 5                 |
| Disruption minimization | “... results in zero disruptions.”                          | 4                 |
| Skill coverage          | “... ensure there is always someone with the network skill” | 3                 |
| Solver reliance         | “... arrangement calculated by the computer”                | 2                 |

Table 9.1: Qualitative Codebook for Task 1 Textual Explanations

The qualitative analysis demonstrates that participants primarily evaluated whether workers possessed similar or distinct skills. This indicates that participants conceptualized the problem through the connections and overlaps between workers, supporting the premise that linkage learning provides a valuable explanatory framework.

Participants were subsequently asked about their confidence and perceived ease during this task:

- “I felt confident in the explanation I gave to the employee”: 4 Somewhat confident, 1 Extremely confident, 1 Neutral, 1 Extremely not confident.
- “I am confident in the ordering I chose”: 4 Agree, 1 Strongly Agree, 2 Neutral.
- “I understand why the changes were disruptive”: 5 Agree, 1 Strongly Agree, 1 Neutral.
- “I found it easy to choose the ordering”: 2 Agree, 3 Neutral, 2 Disagree.

Participants generally understood the task mechanics and were confident in their final decisions. However, opinions on task difficulty varied, suggesting the process still required noticeable cognitive effort.

Regarding the explanations specifically:

- “The explanations were helpful”: 5 Agree, 1 Strongly Agree, 1 Disagree.
- “The explanations were easy to understand”: 6 Agree, 1 Disagree.
- “I believe the explanations were accurate”: 6 Agree, 1 Strongly Agree.
- “The explanations made me trust the solution more”: 5 Agree, 1 Strongly Agree, 1 Neutral.

The provided materials successfully fostered algorithmic reliance; participants found them accurate, understandable, and effective at increasing trust in the system’s output.

| Theme                   | Example Quote                                                                         | Freq. ( $N = 8$ ) |
|-------------------------|---------------------------------------------------------------------------------------|-------------------|
| Disruption minimization | “...resulted in the greatest amount of disruptions since they do not share skills...” | 7                 |
| Skill coverage          | “...there being a day with out someone for either diagnostics or maintenance.”        | 5                 |
| Skill sharing           | “Because Brandon and Kevin have overlapping skills...”                                | 4                 |

Table 9.2: Qualitative codebook for participants’ long-form answers in Task 2 (without explanations).

#### 9.4.1.5 Analysis of Task 2 (without explanations)

In the second task, participants were again asked to write a textual explanation for the affected workers, but without access to algorithmic explanations. The resulting codebook is presented in Table 9.2. The recurring themes remained consistent with the first task, though “Disruption minimization” (i.e., the objective fitness) was cited more frequently. This suggests that without explanatory support, participants focused directly on evaluating the raw objective function rather than relying on underlying constraint concepts.

Participants reported their confidence as follows:

- “I am confident in the choice I made”: 4 Agree, 2 Strongly Agree, 2 Neutral.
- “I have reasons for my choice”: 5 Agree, 2 Strongly Agree, 1 Neutral.
- “I felt confident in the explanation I gave to the employee”: 4 Somewhat confident, 2 Neutral, 1 Very confident, 1 Extremely not confident.

Even in the absence of algorithmic explanations, participants felt justified in their choices. However, their confidence in explicitly explaining those decisions to employees was more reserved, clustering around “Somewhat confident” and “Neutral.” This implies that while participants trusted their own intuition or manual problem-solving, they found it difficult to articulate an authoritative rationale without the backing of explanatory materials.

#### 9.4.2 *(Hypothetical) Quantitative analysis of Task 1*

Since the participants were required to rank a list of possible modifications to the optima. To evaluate this, it is compared against the ground truth ordering (based on fitness) by using Spearman’s Footrule to determine its distance from the best answer [79]. Finally, these distances are compared between the groups using an independent-samples t-test to determine if there is a statistically significant difference between the groups.

#### 9.4.3 *(Hypothetical) Quantitative analysis of Task 2*

For this task, the user created a new solution by selecting 2 workers (out of 5) who would like to have their working pattern changed. The fitness of this solution is used to score the user’s performance, which can be compared by using a simple t-test between the control group and the group with explanations.

This approach proved useful in technically giving the user plenty of options while still posing the question succinctly.

### 9.5 ON THE INCONCLUSIVE OUTCOME OF THE STUDY

The data obtained from our user study was not usable for any concrete evidence of our explanations being understandable and actionable. This stems from an oversight in experimental design: there needed to be an incentive for the user to perform well. In our case, the participants received a monetary reward after completing the questionnaire, and so many of the users completed it as fast as possible by giving random answers. Resolving this issue is complicated by university policies, which prevent the rewards of user studies from being dependent on participants’ performance.

The lack of data was also caused by random chance: if the bad-faith users had been assigned more evenly between questionnaires (which was a random process), then the groups would have been more balanced.

Future extensions of this thesis will endeavour to complete this user study, by finding participants who are expected to provide usable results.

### 9.6 CONCLUSION

This chapter outlined the design and deployment of a user study to evaluate our novel kind of explanations. Considering the available literature on XAI evaluation and possible incompatibilities with metaheuristics, we developed a user study which assessed the users via both objective and subjective (self-reported) metrics. Preliminary results show that the users are wary of using automated methods, but are satisfied with our explanations. Quantitative information and more conclusive results are not available.

**Future work** will involve the completion of data collection, and a full assessment of the effect of our explanations.

## TESTIMONIAL FROM BRITISH TELECOM RESEARCHER

This chapter aims to provide an indication of to what degree the methodology presented in this thesis satisfies the original research intentions presented by BT.

Our industrial point of contact, Dr. Russell Ainslie, was interviewed on the exact problems faced at BT, and how much our methods can help with them. The 2-hour interview made use of an interactive tool which simulated the workflow of a manager at BT assigning rotas to engineers, which included our explainability tools. The presence of this demo was included so that the testimonial given is from hands-on experience, as a means to reduce confounding effects such as the illusion of understanding.

The interview involved the following stages:

1. Questions on Dr. Ainslie's background and experience with the Staff Rostering problem
2. Demonstration of the tool (metaheuristic to solve the problem, solution viewer, solution editor) **without explanations**.
3. Task: Rota modification **without** explanations available
4. Questions on the main challenges in solving the task
5. Task: Rota modification **with** explanations available
6. Another task without explanations, and then with explanations.
7. Questions on usefulness of explanations provided
8. Questions on suitability of the testimonial

These stages will be reported in the following sections, but as a summary:

- Dr. Ainslie is not a manager that would usually be involved in the rota-assignment tasks, but he has sufficient experience in working with them
- Dr. Ainslie indicated that the SR problem is only one of some optimisation problems at BT which would benefit from explanations. The generality of our explanations is indicated to be useful in adopting them.
- The challenges of the SR problem stem from the quantity of variables, their interactions and associated tables of data
- The explainability tools provided were used heavily in the process of attempting to modify solutions. Linkage learning was used directly for insight (which was not expected), and descriptors were used as a guide for what each PS does rather than looking at the patterns themselves.
- The PSs did not always lead to improved results in terms of fitness, but they did make the tasks less taxing and time-consuming
- Our explanations offered useful insight, and he did try to use different objectives and different trade-offs between them. Applying constraints to the PS mining process was the most used feature.
- In many cases, following the advice given by PSs did not lead to the desired result. The PSs provided did have a great impact on fitness when applied on the best solution, but it was unpredictable whether this was a positive or negative change.
- The explanations provided are not able to suggest what problem settings would be changed, such as which new employees should be hired.

In general, our industrial collaborator agrees that the explanations provided align with the objectives of the research.

## 10.1 INTERVIEWEE BACKGROUND AND CONTEXT

To establish the context of the testimonial, the interview began by defining Dr. Ainslie’s relationship with the **SR** problem. While not an end-user or rota manager himself, his research team was responsible for creating the algorithms and tools intended to assist manual planners. The Staff Rostering task was originally selected as a proof of concept for explainability because it represented the most accessible tier of resource optimisation at **BT**, with the intention of eventually adapting the explainability tools to more complex scheduling and planning problems.

The motivation for explanations is the pervasive mistrust of black-box optimisation among end-users (workers, HR staff and managers). **Dr. Ainslie confirmed that there is reluctance from users to blindly accept algorithmic output, even when indicators of quality are provided.**

### 10.1.1 *Black-box metaheuristics are necessary*

Black-box metaheuristics are necessary to produce satisfactory solutions for Staff Rostering. Managers in the past manually created solutions, and these arrangements also lent themselves better to explanations, but the solutions were low quality compared to what a metaheuristic can achieve. Dr. Ainslie explained how it is **impossible to manually obtain high-quality rota assignments**: human users simply are not able to deal with the quantity of data available, the linkage effects present and the fitness metric used.

### 10.1.2 *Black-box metaheuristics are not trusted*

When asked “Do humans trust that automated optimisation solvers produce results that meet operational requirements?”, Dr. Ainslie indicated that this is a gray area. Researchers trust the methods they work with (even black box methods), but the end users (workers, HR) do not.

When asked about the causes of the mistrust, the following possible causes were presented:

- Users do not trust that the automated system actually performs well;
- Users anthropomorphise the metaheuristic, and think that its decisions are based on “priorities” which do not match human priorities.

In either case, the consequence is that the solutions are either created manually or altered manually, both of which have a high chance of producing less desirable solutions.

When asked “When understanding a solution, what information would you benefit from that is currently missing?”, his answer can be summarised as *contrastive explanations*: why a certain worker was assigned to a rota rather than another. Another aspect mentioned is understanding the links between workers: since the problem is complex due to rota overlaps and shared skillsets, it is difficult to determine which workers are linked. This last part is particularly relevant to our work, since in this thesis it was argued that **linkage effects are the main contributor to problem complexity**. Throughout this interview, linkage was a recurring topic.

## 10.2 TASK WITHOUT EXPLANATIONS

Following from the questions relating to the Staff Rostering problem, Dr. Ainslie was shown the interactive demo, implemented as a website. The website implements an instance of the Staff Rostering problem (the one used throughout this work, such as in [46]), solves it using iRoster, and presents the best solution found.

The tools available to Dr. Ainslie were:

- A best solution viewer (Figure 10.1), where various aspects (breakdown of fitness function in its terms, rotas, skill sets) of the solution and the fitness function are tabulated.
- A solution modification tool (Figure 10.2), where the user was able to apply last minute changes to the solution and view their effect on fitness.
- A “random-fault” generator, which produced an unforeseen constraint: an employee is not able to work on their assigned rota. This was used to mimic scenarios in



Figure 10.1: Best solution viewer tools, available to our industrial collaborator during the interview. The fitness function is broken into its summands (range scores for every combination of skill and weekday), and these are shown as a heatmap (top of figure). These range scores are based on how many workers with each skill appear throughout the 3 month period, as shown in the mid-section of the figure. The rota assignments for each worker, along with other rota patterns available and skillset, are shown in the bottom-left of the figure. The user was able to compare rota patterns (bottom-right).

### Manual Modification Sandbox

Select Worker:  
Employee#40 (Cur: Opt #1)

Assign Alternative Rota:  
Option #4 (Pref 4)

Add to Modifications
Reset All Mods

#### Active Modifications List

| Worker     | Original | Modified | Isolated Delta | Action |
|------------|----------|----------|----------------|--------|
| Worker #31 | Opt #0   | Opt i    | -10.8688       | Remove |
| Worker #46 | Opt #1   | Opt i    | -3.7450        | Remove |
| Worker #40 | Opt #1   | Opt i    | -0.9471        | Remove |

#### Compound Fitness Impact

Baseline Solution Fitness: -287.7142  
Compound Modified Fitness: -297.3465  
Compound Net Delta: -9.6323

Figure 10.2: Solution modification tool. This allowed the user to apply small modifications to the best known solution, and view the effect on fitness. The user could select multiple workers and change each of their rotas. The effect (fitness change) of each individual change is shown on the left, and the overall change in fitness (subject to linkage) is shown on the right.

### Emergency Leave Simulator

Simulate a scenario where a worker unexpectedly takes leave and must be reassigned to an alternative rota.

Simulate New Leave Scenario

#### Incident Report

**Worker:** Employee#9 (Employee\_009)  
**Reason for Absence:** Sprained Ankle  
**Current Assignment:** Option #2 (RP\_11)  
**Baseline Fitness:** -287.7142

#### Alternative Rota Impacts (Read-Only)

| Alternative Rota Option | Preference Rank | Simulated Fitness | Fitness Delta |
|-------------------------|-----------------|-------------------|---------------|
| Option #0 (RP_22)       | 0               | -287.7434         | -0.0292       |
| Option #1 (RP_63)       | 1               | -287.7320         | -0.0178       |
| Option #3 (RP_82)       | 3               | -287.7444         | -0.0302       |

Figure 10.3: Last-minute emergency tool. This produced a random emergency, where an employee was not able to work on the assigned rota pattern.

which a manager would have had to manually modify the rota on the fly (using the solution modification tool).

During the first practical task, Dr. Ainslie was asked to manually modify a roster to accommodate a sudden worker absence (simulating a real-world disruption, generated using the random-fault generator to avoid interviewer bias). The modification was to be applied using the tool provided, which also showed the resulting fitness value. Importantly, **at this stage he was not shown the explainability tools**.

Two separate “emergencies” were generated, and then the interview process continued, focusing on the complexity of manually modifying solutions. This initial task gave Dr. Ainslie a chance at experiencing first-hand what this process would be like, which was especially important given that he is not directly involved in this process otherwise.

#### 10.2.1 *The interface*

A potential bias of this testimonial was the possibility that the interface and tools (without explanations) were not on par with what would be provided at BT. Dr. Ainslie indicated that the tool was useful in his opinion, and that it is more comprehensive than what is provided at BT. The break-down of the fitness function as a combination of ranges, coloured as a heatmap, is not shown to BT managers.

#### 10.2.2 *The role of fitness*

The fitness function was an important component of the tasks, as our industrial collaborator was tasked to use that as a metric of solution quality. The interactive interface showed would-be fitness values when the solution was modified, but it was unclear if this was accurately reflecting real-life manager scenarios. To clarify this, Dr. Ainslie was asked “In a real-life scenario, would fitness be known when the solution is being manually changed?”, to which he replied that it depends. Some employees have tools used to ensure the rotas follow legal constraints and these also show the fitness values. In other cases, the solutions are being modified without regard for the fitness function. In either case, he clarified that it would not be called “fitness”, but its underlying meaning would be the same.

Moreover, we wanted to ensure that the fitness function in use was accurate to what would be used in real life as an indicator of rota assignment quality. He assured that the fitness used for this demonstration (which is the same as what was used in this thesis) is accurate to what would be used at BT.

#### 10.2.3 *Task completion complexity*

Dr. Ainslie was asked “Using the tools available, how intuitive / easy is it to identify which parts of the solution are the main drivers behind fitness?”. His reply was that while tools can be useful, it is still unfeasibly time consuming to properly understand the fitness function.

This will be a recurring theme throughout this chapter: the Staff Rostering problem does not lend itself to human-manipulation because of the quantity of data available. Individual data (such as a worker) is simple to understand, but the relationships between multiple elements leads to emergent complexity.

Dr. Ainslie was asked about his strategy in completing the tasks, to which he described how he first tried to find workers with similar skills and rota patterns. If an employee was not able to work on their rota, he would try to find a “replacement” that could work on that rota.

He described how this process was complex and error-prone, because the effects of rota changes were unpredictable. He also noted how this process would be even harder if the fitness function was not known.

#### 10.2.4 *Trust and human explanations for manual modifications*

Dr. Ainslie was then asked questions relating to trust, either from himself in his own actions in the given tasks, or from hypothetical stakeholders.

The answers are summarised as follows:

- **Do you trust that your decisions were relatively good compared to other small changes that could have been done?** He did not trust that his changes were desirable;
- **Were you confident in your ability to explain *why* you made certain decisions?** He did not feel confident, especially since he described his process as “mostly random until something good is found”.
- **If you were to give explanations, would they be accurate?** He answered that he could not give an accurate explanation.
- **Would the affected workers and HR trust your explanations?** Although in general the answer is “no”, he also mentioned how many employees could be deceived into believing a researcher like him if a sufficiently confident presentation was adopted.

Considering these answers, it can be concluded that manually modifying rota assignments leads to many sources of mistrust, and that presence of accurate explanations is hard if not impossible.

### 10.3 TASK WITH EXPLANATIONS (LINKAGE LEARNING AND PARTIAL SOLUTIONS)

In the subsequent task, the explainability tools - **PS** mining but also linkage learning - were made available to our industrial collaborator.

The tools available were as follows:

- A Linkage Learning viewer (Figure 10.4), where the user could run a linkage learning method and view the results as a heatmap and a browsable table. Using this, a user can see what workers are linked to every other worker.
- **PS** mining configuration (Figure 10.5). Here the user could start a **PS** mining process, with the results being shown on another page. The user had full control of all the aspects of our methodology, including the objectives.
- **PS** viewer (Figure 10.6), where the user was shown the Pareto Front found by the **PS** mining process. For every **PS**, the user could also choose to generate its descriptors (with settings available to the user as well). The user was also able to cluster the **PSs**, and order them as they desired, although minmax regret was the default view.

#### 10.3.1 Task strategy with explanations

Dr. Ainslie was asked about his strategy when explainability tools were available, and how he used them. Of the tools we developed, he mostly made use of the **PS** viewer, where he described how after a single **PS** mining run he would browse through **PSs** to find a suitable one to use as a “patch” for his problem. He also indicated that this process was heavily guided by the descriptors of **PSs**.

The configuration of **PSs** mining was used more in the beginning, but after he settled on certain settings there was no reason for him to adjust them further. The settings he settled on were:

- Explanation scope: Local (i.e. The search space used was **Subset-of-Solution**);
- **PS** evaluation budget: 6000;
- **PS** population size: 100;
- objectives: **\*-count simplicity, mean-fitness and atomicity+independence**;

The main setting to note was that instead of using traditional atomicity, he chose to use **atomicity+independence**. This choice was motivated by the need to find “patches” (groups of workers and rota assignments) that are self-contained rather than being dependent on unknown decision variables (i.e. the rest of the solution).

This is an important result from this interview, as it shows how allowing the user to select their objectives and settings is both approachable to a user, and allows them to tune the system to their preferences.



Figure 10.4: Linkage learning viewer. The user was able to decide which linkage learning method to use (Perturbation by default), and view the resulting linkages as a heatmap (middle section of figure). The user was also able to select individual workers, and see which other workers they were linked to (bottom section of figure).

### Multi-Objective Partial Solution Mining (NSGA-II)

Search Scope:

Subset-Of-Solution (Local Explanation of Best)

Evaluation Budget:

6000

Population Size:

60

Pattern Constraints

Constraint Type:

No Constraint

Active Objectives (Minimization):

☒ Mean Fitness
 ☐ MWU Test P-Value
 ☒ Atomicity
 ☐ Atomicity + Independence
 ☒ Simplicity
 ☐ Inverse Simplicity
 ☐ Independence
 ☐ Robustness
 ☐ Sample Count
 ☐ Inverse Sample Count

Run PS Mining

Status: Idle.

Figure 10.5: PS mining settings selection interface. This allowed the user to perform a PS mining search with the desired fitness evaluation budget, population size, search space (local or global explanations), objectives and constraints). The user could enforce (via the constraints) that the pattern needed to assign certain workers a given rota, or to avoid a given rota.

### Manual Modification Sandbox

Select Worker:

Employee#40 (Cur: Opt #1)

Add to Modifications

Reset All Mods

Assign Alternative Rota:

Option #4 (Pref 4)

#### Active Modifications List

| Worker     | Original | Modified | Isolated Delta | Action |
|------------|----------|----------|----------------|--------|
| Worker #31 | Opt #0   | Opt #1   | -10.8688       | Remove |
| Worker #46 | Opt #1   | Opt #1   | -3.7450        | Remove |
| Worker #40 | Opt #1   | Opt #1   | -0.9471        | Remove |

#### Compound Fitness Impact

Baseline Solution Fitness: -287.7142

Compound Modified Fitness: -297.3465

Compound Net Delta: -9.6323

Figure 10.6: Partial Solution viewer. This tool allowed to view the PSs found by the PS mining process. The PSs are ordered according to the criterion selected by the user (minmax regret by default, other options were the objectives used for the PS search). The user could also optionally cluster the items shown. Each PS could be selected to show its pattern, and descriptors could be generated by clicking on “Calculate descriptors”. The settings used for calculating descriptors were selectable by the user (top of figure, by default the descriptor threshold was 0.1, and the quantity of samples was 1000).

Finally, he mentioned how he did not use the linkage learning tools directly, as he found the **PSs** to be more directly related to his intentions. Although he did not use linkage tools directly, he did use it indirectly because of the **PS** objectives he chose, and in fact this was one of the main tools he used.

### 10.3.2 *Interface user experience*

To ensure that the overall interface of explanations was not detrimental, he was again asked if the interface affected his performance in solving the tasks. He used the terms “useful” and “handy” in relation to the **PS** viewer specifically, and he was satisfied with what was provided.

He was then asked about how understandable were the explanations provided, and he indicated that it was all easy to understand, although the concept of descriptors required more explaining from the interviewer.

He also mentioned how the cognitive effort required in using the explanations was lowered thanks to the layout and colour coding.

An aspect which was important in the original research intentions was for the explanations to be generated efficiently, so that managers are still willing to use them. We asked our industrial collaborator if our implementation is efficient enough, to which he replied positively, noting that even when implemented in a browser it was sufficiently fast.

### 10.3.3 *Advantage of explanations*

Dr. Ainslie was asked about whether the explainability tools provided made the task faster, cognitively easier and whether it led to better results. He clarified how the tasks were **cognitively easier but also cognitively different**: it was still mostly navigating lists of information, but the format of explanations provided made the process faster and easier to process. The explanations did help our industrial collaborator in completing the tasks, but this was not always a linear process. Unfortunately, the explanations provided sometimes led to undesirable results (i.e. fitness could get much worse rather than getting better). Dr. Ainslie blamed himself for this, but the issue should be attributed to the explainability system: there should be clear indicators of which **PSs** would be most suitable as “patches” with positive effects. This issue stems from our system not matching Dr. Ainslie’s mental model (which is an issue described in [192]), and in future work it should be investigated how this can be remedied.

When asked whether the explanations present were hard to understand, he reported that they were clear mainly due to their colour coding and the descriptors available. He added that descriptors helped him navigate the quantity of **PSs** presented. Following from this, he was asked if the quantity of data presented was overwhelming, to which he replied negatively.

The quantity of information provided is not overwhelming, but still takes time to parse, but tools such as descriptors help in removing attrition.

Dr. Ainslie was then asked whether he felt in control of the process, of whether he was passively accepting them (an issue described in [178]). He replied that he felt reasonably in control, and that he was the one making decisions on what explanations to generate and to consider, and ultimately in modifying solutions.

Although he was satisfied with the explanations, we also wanted to know whether he thought **BT** would benefit from our methodology, in various aspects. He answered “yes” to all of the following:

- solving tasks more quickly;
- solving tasks more easily;
- training new employees more quickly;
- getting better results;
- reduce barriers to adoption;

Finally, he was asked whether “Partial Solutions are effective in supporting decision making”, to which he answered positively, and that his decisions were grounded on the explanations provided.

### 10.3.4 *Trust and human explanations for manual modifications*

Similarly to before, Dr. Ainslie was asked questions relating to trust and accuracy of his understanding.

When asked “How much do you trust that your decisions were relatively good compared to other changes that could have been done?”, this time he explained how it is still not “an exact science”, but it is still better than random (which was the strategy when explanations had not been provided).

Furthermore, he was asked if he would be able to confidently explain and justify his decision to HR and affected workers. To this, he answered that **since he had more of a process, especially in selecting groups of linked workers, his thought process was easier to explain to HR and affected workers**. He described his hypothetical justifications as “accurate enough”, and how the audience of his explanations would understand them due to “linked workers” being a familiar idea.

## 10.4 MISSING ELEMENTS

In order to get a more balanced opinion, we also asked our industrial collaborator the following questions:

- **“Did the problem, scenarios shown, and given tasks match what is experienced at BT?”**
- **“What aspects of the explanations did *NOT* cover, from BT’s perspective?”**
- **“Do you think that this system [our methodology] can be applied beyond the Staff Rostering problem?”**

To these questions, he answered that the scenarios given are accurate, and that he was confident that the same methods can be applied on other real-world problems.

At the same time, he also said that there are still issues which are not resolved by our explanations:

- **fairness:** the explanations do not show whether there is a bias in how the metaheuristic assigns certain workers;
- **problem modification:** the explanations do not suggest how the problem could be modified to produce a better optimal solution.

These aspects may be tackled in future research, but it should also be acknowledged that there cannot be an explanation method that can fulfil all purposes (as discussed in [7]).

### 10.4.1 *Conclusions from Testimonial*

In concluding his testimonial, Dr. Ainslie confirmed that the explanations provided by the PSs were actionable, transparent, and generated quickly enough for practical, real-time use within BT’s operational constraints. He confirmed that providing this level of explainability will help solve tasks more easily and reduce user mistrust towards metaheuristics. The explanations were not perfect: they sometimes resulted in undesirable outcomes. This indicates that while the system has achieved its purposes, there is still room for improvement.

Regarding wider applicability, Dr. Ainslie concluded that the methodology demonstrated in this thesis is not strictly limited to the SR problem, and has strong potential for broader applications across BT’s operations.

Part IV

## CONCLUSION

This chapter concludes the thesis by summarising the main findings and evaluating the proposed methodology. It revisits the research questions established in the introduction to assess how successfully they have been answered, and provides a critical discussion of the advantages and limitations of the developed frameworks. The chapter closes with possible future work.

### 11.1 ANSWERS TO THE RESEARCH QUESTIONS

The primary aim of this thesis was to investigate methods for making metaheuristic solutions more transparent. This was broken down into five research questions, which have been addressed throughout the contribution chapters.

**RQ1: HOW CAN THE RESULTS OF A METAHEURISTIC BE EXPLAINED?** This thesis shifts the focus from the metaheuristic to the solutions themselves. The metaheuristic is a tool to find a near-optimal solution, but it generates additional data which can be used as **background information** for the best solution found (the  $P_{\text{Ref}}$ ).

The explanations are based on the concept of **PSs** (defined in Chapter 5), which are optimal patterns with respect to certain objectives (defined in Chapter 6). These **PSs** can then be post-processed (Section 5.6) into human-readable explanations. They can be used as local explanations (for a single solution), global explanations (relating to a specific dataset), and within models (Section 8.3).

**RQ2: HOW CAN PARTIAL SOLUTIONS BE DEFINED AND GENERATED IN ORDER TO BE USABLE FOR SATISFACTORY EXPLANATIONS?** **PSs** can be used for satisfactory explanations when they satisfy the following objectives (Chapter 6).

In the general use case, desirable **PSs** are associated with high fitness, simple and atomic. It is important to define each of these in detail, but also to consider them at the same time as a multi-objective problem in order to account for their trade-offs.

For example, there are many ways to define what makes a pattern “associated with high fitness”, which are described in Section 6.7 - from simply measuring their average fitness to various statistical tests. Users who require guarantees of the patterns having provably higher average fitness should use the latter.

There are many motivations for explanations, and therefore there should be many objectives available. One motivation which was used for this thesis was the ability to modify solutions to fit unforeseen constraints. In order to test if **PSs**-based explanations assist this process, a user study was designed and executed, but this test will need to be repeated in future work in order to obtain conclusive results.

**PSs** are extracted using a multi-objective search process (NSGA-II) equipped with customised search operators (defined in Chapter 6). By operating on a static dataset of evaluated solutions ( $P_{\text{Ref}}$ ), the mining process remains independent of the original metaheuristic solver. The objectives were developed to remain efficient, and to allow the system to run in the order of seconds and minutes rather than hours. On problems of sizes up to 100 variables, our methods are below the 2 minute threshold for interactive use, as shown in Section 6.14.

**RQ3: HOW EFFECTIVE ARE PARTIAL SOLUTIONS IN SUPPORTING DECISION MAKING?** Although the preliminary results of our user study (Chapter 9) indicate that the users found the explanations useful, more quantitative results can only be obtained by repeated the study at a bigger scale. The explanations produced in many scenarios were discussed in relation to their practical use, with Chapter 8 containing many discussions of how they can be used for practical purposes. For example, in Section 2.3.2 it was shown how the explanations can inform managerial decisions and explain to affected workers the decisions taken by the automated method.

**RQ4: CAN PARTIAL SOLUTIONS BE USED AS MODELS TO PREDICT FITNESS IN AN EXPLAINABLE MANNER?** By integrating **PSs** into decision

trees, we created novel, transparent models capable of predicting solution fitness (Section 8.3). These PS-based models proved to be reliable predictors (as good as traditional decision trees) while remaining highly interpretable to end-users. Although the predictive performance was satisfactory, it was unclear if the models were useful as explanations. The user study designed in [42] is work towards proving this quantitatively, but it will need to be continued in order to get statistically valid results.

**RQ5: CAN PARTIAL SOLUTIONS BE USED IN OTHER DOMAINS, SUCH AS RECOMMENDER SYSTEMS?** The general applicability of the framework allows it to transition smoothly to other domains. We successfully applied PS mining to Recommender Systems datasets (Section 8.2). The methodology proved capable of finding and explaining actionable patterns even in the absence of a traditional search method.

**WITHIN THE HYPOTHESES** The answers given above to the research questions can now be used to answer the hypotheses presented in Chapter 1:

We hypothesise that solutions produced by metaheuristics for combinatorial optimisation problems can be made more understandable (**RQ1**). This is achieved by breaking them into smaller patterns, which we call Partial Solutions (**RQ2**). The fitness of a solution can be understood as the combined effects of the Partial Solutions present (**RQ4**).

We hypothesise that it is possible to find Partial Solutions associated with high fitness which are also understandable (**RQ2**). These Partial Solutions can then be used for explanations (**RQ2 and evaluated in RQ3**). Partial Solutions can also be used to implement transparent prediction models (**RQ4**), but also analyse datasets such as those produced by a Recommender System (**RQ5**).

## 11.2 DISCUSSION OF ADVANTAGES AND DISADVANTAGES

Naturally, there are both advantages and disadvantages to the methods proposed in this thesis.

**ADVANTAGES** One of the design principles behind the frameworks developed was to keep the methodology very general, so that it may be adapted for new unforeseen uses. This is seen in the following aspects:

- The use of a static dataset ( $P_{\text{Ref}}$ ), so that our methods do not need to be re-implemented for any metaheuristic to be analysed.
- A wide range of objectives, which are not dependent on the optimisation problem or the metaheuristic used.
- Linkage learning methods which are designed to be robust against uneven samples within the  $P_{\text{Ref}}$ .
- Using a Multi-objective method, rather than settling on a specific trade-off between the objectives. The search operators themselves are also designed to be general.

These points allowed for PSs to be applied in two contexts which were not expected when PS mining was first conceptualised: the creation of PS-based models, and the explanations seen in Recommender Systems.

Individual components of this research - linkage learning and the various objectives defined for PSs - are useful in their own right.

**DISADVANTAGES** The core of our framework is a stochastic process, and this leads to unavoidable uncertainty in the results. It is unclear if a PS mining run is producing the absolute best patterns for the user, for example.

Moreover, the explanations produced are always shown as a trade-off between certain objectives. The user might not have any interest in these trade-offs, or the time to understand them. We argue that this is a necessary level of nuance to produce valid explanations, as opposed to unprovable or deceptive ones.

The search process itself is not as efficient as a highly specialised heuristic method could be. This is the direct result of prioritising flexibility over efficiency. Despite this, the

system has been designed to scale as the size of the  $P_{\text{Ref}}$  and the solution size increases. While it is definitely possible to develop a faster process for **PS** mining, empirical evaluations demonstrated that **PS** mining generally satisfies the two-minute execution threshold requested by our industrial partner, **BT**. Further optimisations could reduce this computational overhead, but current speeds remain viable for practical, interactive use.

### 11.3 FUTURE WORK

This section outlines promising directions for extending the methodology developed in this thesis. While the current approach successfully mines **PSs** to explain combinatorial optimisation problems, its core principles can be generalised. The following sections detail five primary avenues for future work: proving the utility of our methods by continuing the user study proposed in [42], expanding to continuous search spaces, applying **PS** mining to broader machine learning contexts, leveraging **PSs** to guide search performance via a novel **EDA**, and integrating **PS** generation into Learning Classifier Systems to improve rule inference.

#### 11.3.1 *Continuing the user study*

**XAI** methods often lack evaluation on human users, meaning that there is no proof for their usefulness. Our work suffers from the same issue; however, an attempt was made to measure the advantage brought by our explanations in the user study described in [42]. Due to a lack of usable user data, the study’s results are inconclusive, and therefore future research should continue it. The value of this user study is not only to prove that our methods achieve their aims, but also to provide an evaluation methodology which specifically targets **XAI** for metaheuristics.

#### 11.3.2 *Expanding applicability to continuous and other kinds of problems*

The main limitation of our methodology is that it is only applicable to combinatorial optimisation problems.

In general terms, a **PS** needs to be an entity that is either “present” or “absent” from a solution (more formally a **predicate**). Many components of our methodology work entirely from this assumption, and thus they would not need to be changed:

- Mean-fitness: simply the average of matched solutions
- Descriptors: based on the average of proxy-fitness values in matched solutions

Future research which generalises **PSs** will be able to re-use most of the methodology shown in this thesis. What remains to be discussed is how exactly can we generalise **PSs**?

**DEFINING PARTIAL SOLUTIONS FOR CONTINUOUS SEARCH SPACES**  
There are many possible ways in which **PSs** can be defined for continuous spaces, which can be borrowed from Learning Classifier System research. A comprehensive survey is given in [244], with some representative examples being:

- **Interval-based representations:** each variable is either  $*$  or an axis-aligned range. This is easy to implement, but it does not fit oblique decision boundaries.
- **Ellipsoidal-based representation:** the region of the matched search space is the overlap of multi-dimensional ellipsoids. These rulesets involve more parameters to evolve, and have a tendency to need simplification heuristics.
- **Convex hull:** these are a set of points which define a convex region of the search space. Their representation allows for faster convergence. The search process required is more complex than the two methods above, and the number of points required grows exponentially with dimensionality.
- **GP-like representation:** GP-trees are evolved, which have the potential of being easily understood but also to represent a wide range of patterns.

The last approach is likely to be the one which would be pursued, because it has the greatest generality and has been experimented upon by Poli et al as part of their

research on schemata for GP-trees [286]. GP-trees could even work on combinatorial and continuous search spaces with little modification. A tree would also be able to represent higher-order interactions, such as “if variable A and variable B are *different*”, which would allow the evolved rules to be much richer. A final advantage of these GP-trees would be the large body of literature relating to their evolution: many operators and search objectives are already defined, with some already focusing on understandability [277, 67, 23].

### 11.3.3 *Expanding applicability to machine learning*

Throughout this thesis, the use of variable importance as a means to understand machine learning models has been criticised because it ignores variable interactions. Our work is already usable as a replacement for these methods, which can be seen in Section 8.2.

Another use would be in hyperparameter tuning: for each combination of hyperparameters, there is a scalar representing the predictive performance of the model. Our methodology can be applied in order to determine if certain hyperparameters should be considered together. Although the aforementioned approach would work, it could be specialised to implement the following features:

- Determine kinds of inputs for which a model performs poorly or is biased
- Determine kinds of inputs which are under-represented in the data
- Assist feature selection by deciding a set of highly-linked features
- Evolve patterns which are small but disruptive when applied onto inputs

These scenarios can be implemented simply by creating new objectives that make use of the ground truth value for each input, as well as the predicted values. Much of this methodology could borrow from research on adversarial attacks for classification and regression tasks [18].

### 11.3.4 *Partial Solutions for search performance: a PS-based Estimation of Distribution Algorithm*

If **PS**s are indicators of high fitness in solutions, it is easy to see how they could be used within a search method for optimisation. A small attempt at this was already implemented in [44] (attached in Chapter A), where a solution is progressively built by repeatedly picking patterns from a collection.

A more sophisticated approach could take the shape of an **EDA**, which would have the advantage of being more understandable than a typical one. An **EDA** requires the following mechanisms:

- Exploration: generating solutions which are different from those found already
- Learning: inferring a model representing solutions with high fitness
- Exploitation: generating solutions with a high expected fitness

These mechanisms could be implemented by creating two collections of patterns:

- The first collection contains patterns which have high mean-fitness. This is identical to the methodology used throughout this thesis
- The second collection contains patterns which are under-represented in currently evaluated solutions.

Each of these collections is found via **PS** mining, but with different objectives. Patterns can then be picked from these collections to build new solutions. In the exploration stage, the second collection should be picked more often, whereas in the latter stages, the exploitation patterns are used more. The solutions are then used to update and refine the collections of patterns, and the patterns are used to expand the collection of solutions. The creation of every solution then has a simple explanation: a combination of patterns, which the user should be able to understand.

### 11.3.5 *A Human-in-the-loop optimiser based on PSs*

If explanations can be considered a tool to assist trial and error (as described in Section 3.3), then it is easy to see how PSs could be used to implement a HITL search method.

A population-based metaheuristic can be hard to control for a human due to the multitude of solutions to be managed, but PSs offer a simple mechanism to both describe and take action on *groups of* solutions.

At every generation, the population may be summarised in terms of the PSs present, and the user can decide which patterns are worth investigating and which are likely due to *hitch-hiking*. This feedback then translates into selection, mutation and crossover operators so that the next generation is *caused* by the PSs rather than being explained by them. Therefore, such an algorithm would be closer to being transparent than a traditional metaheuristic.

### 11.3.6 *A PS-based Learning Classifier System*

Learning Classifier Systems already use rulesets that are similar in structure to PSs, but they are generated using specialised stochastic heuristics. These heuristics are known to converge slowly [271], and indirectly favour rules which are simpler. Our methodology offers a more comprehensive method of inferring the rules (using simplicity, atomicity or independence for example), which could improve the creation process behind Learning Classifier Systems. Our positive results on decision trees ([45]) give an indication that this idea should work, as PSs within decision trees proved to be a reliable predictor of solution fitness.

## 11.4 FINAL REMARKS

Explainability in Evolutionary Computation is not just an academic curiosity. As metaheuristics are increasingly deployed in critical, real-world applications, the black-box paradigm becomes unsustainable. Industrial partners require solutions that their human stakeholders can verify, trust, and comfortably deploy.

The research presented in this thesis provides a foundational step towards explainable optimisation. By shifting the focus from explaining the mechanics of the search to explaining the solutions themselves, we offer a practical toolset for users. It is hoped that the concept of Partial Solutions will encourage further developments in the field. Making search methods inherently understandable is crucial for bridging the gap between complex algorithmic solvers and human decision-makers.

## BIBLIOGRAPHY

---

- [1] Bilal H Abed-Alguni. Evomapx: An explainable framework for metaheuristic optimization algorithms. *Expert Systems with Applications*, 298:129514, 2026.
- [2] David Ackley. *A connectionist machine for genetic hillclimbing*, volume 28. Springer science & business media, 2012.
- [3] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 9 2018. ISSN 21693536. doi: 10.1109/ACCESS.2018.2870052.
- [4] Stephen Adams, Tyler Cody, and Peter A Beling. A survey of inverse reinforcement learning. *Artificial Intelligence Review*, 55(6):4307–4346, 2022.
- [5] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS’18, pages 9525–9536, Red Hook, NY, USA, 2018. Curran Associates Inc.
- [6] Bahriye Akay, Dervis Karaboga, and Rustu Akay. A comprehensive survey on optimizing deep learning models by metaheuristics. *Artificial Intelligence Review*, 55(2):829–894, 2022.
- [7] Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M. Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia D’Áz-Rodríguez, and Francisco Herrera. Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. *Information Fusion*, 99:101805, 2023. ISSN 1566-2535. doi: <https://doi.org/10.1016/j.inffus.2023.101805>. URL <https://www.sciencedirect.com/science/article/pii/S1566253523001148>.
- [8] Aylin Alin and SEFA Kurt. Testing non-additivity (interaction) in two-way anova tables with no replication. *Statistical methods in medical research*, 15(1):63–85, 2006.
- [9] Ahmed Alridha, Fouad Alsherify, and Zahir Al-Khafaji. A review of optimization techniques: Applications and comparative analysis. *Iraqi Journal For Computer Science and Mathematics*, 5:122–134, 05 2024. doi: 10.52866/ijcsm.2024.05.02.011.
- [10] André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 2010.
- [11] Plamen P Angelov, Eduardo A Soares, Richard Jiang, Nicholas I Arnold, and Peter M Atkinson. Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(5):e1424, 2021.
- [12] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Ben- netot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.
- [13] Agustin Garcia Asuero, Ana Sayago, and AG González. The correlation coefficient: An overview. *Critical reviews in analytical chemistry*, 36(1):41–59, 2006.
- [14] Jaume Bacardit, Alexander E. I. Brownlee, Stefano Cagnoni, Giovanni Iacca, John McCall, and David Walker. The intersection of evolutionary computation and explainable ai. In *Proceedings of GECCO ’22*, GECCO ’22, pages 1757–1762, New York, NY, USA, 2022. ACM. ISBN 9781450392686. doi: 10.1145/3520304.3533974.

- [15] Thomas Bäck. Self-adaptation in genetic algorithms. In Francisco J. Varela and Paul Bourguine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 263–271, Cambridge, MA, 1992. MIT Press.
- [16] Emilio Rafael Balda, Arash Behboodi, and Rudolf Mathar. Perturbation analysis of learning algorithms: Generation of adversarial examples from classification to regression. *IEEE Transactions on Signal Processing*, 67(23):6078–6091, 2019.
- [17] Shumeet Baluja and Rich Caruana. Removing the genetics from the standard genetic algorithm. In Armand Prieditis and Stuart Russell, editors, *Machine Learning Proceedings 1995*, pages 38–46. Morgan Kaufmann, San Francisco (CA), 1995. ISBN 978-1-55860-377-6. doi: <https://doi.org/10.1016/B978-1-55860-377-6.50014-1>. URL <https://www.sciencedirect.com/science/article/pii/B9781558603776500141>.
- [18] Hubert Baniecki and Przemyslaw Biecek. Adversarial attacks and defenses in explainable artificial intelligence: A survey. *Information Fusion*, 107:102303, 2024.
- [19] Alejandro Barredo Arrieta, Natalia DÁaz-RodrÁguez, Javier Del Ser, Adrien Ben-netot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020. ISSN 1566-2535.
- [20] Roman Bartak. Modelling soft constraints: A survey. *Neural Network World*, 12(5):421–432, 2002.
- [21] Emre Bayamhoğlu. The right to contest automated decisions under the general data protection regulation: Beyond the so-called "right to explanation". *Regulation & Governance*, 16(4):1058–1078, 2022.
- [22] Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.
- [23] Dimitris Bertsimas and Jack Dunn. *Machine learning under a modern optimization lens*. Dynamic Ideas LLC, 2019.
- [24] Nicola Beume, Boris Naujoks, and Michael Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European journal of operational research*, 181(3):1653–1669, 2007.
- [25] Heike A Bischoff-Ferrari, Walter C Willett, JoAnn E Manson, Bess Dawson-Hughes, Markus G Manz, Robert Theiler, Kilian Braendle, Bruno Vellas, René Rizzoli, Reto W Kressig, et al. Combined vitamin d, omega-3 fatty acids, and a simple home exercise program may reduce cancer risk among active adults aged 70 and older: A randomized clinical trial. *Frontiers in aging*, 3:852643, 2022.
- [26] Julian Blank and Kalyanmoy Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020. doi: 10.1109/ACCESS.2020.2990567.
- [27] Christian Blum. Metaheuristics network website. <http://www.metaheuristics.net/>, 2000. Archived at the Internet Archive: <https://web.archive.org/web/20030202120524/http://www.metaheuristics.net/> (accessed September 15, 2025).
- [28] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, September 2003. ISSN 0360-0300. doi: 10.1145/937503.937505. URL <https://doi.org/10.1145/937503.937505>.
- [29] Eric Bonabeau, Carl Anderson, Belinda Orme, Pablo Funes, Sergey Malinchik, Oliver Bandte, Mark Sullivan, and Joseph Rothermich. Methods and systems for interactive evolutionary computing (iec), October 13 2009. US Patent 7,603,326.
- [30] R. D. Boschloo. Raised conditional level of significance for the 2 x 2-table when testing the equality of two probabilities. *Statistica Neerlandica*, 24(1): 1–9, 1970. doi: <https://doi.org/10.1111/j.1467-9574.1970.tb00104.x>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9574.1970.tb00104.x>.

- [31] Peter AN Bosman, Jörn Grah, and Dirk Thierens. Amalgam ideas in noiseless black-box optimization benchmarking. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2247–2254, 2009.
- [32] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [33] Leo Breiman, JH Friedman, RA Olshen, and CJ Stone. Classification and regression trees: Wadsworth international group; 1984. *Citado na*, 24, 1984.
- [34] Andrea Brennen. What do people really want when they say they want "explainable ai?" we asked 60 stakeholders. In *Extended abstracts of the 2020 CHI conference on human factors in computing systems*, pages 1–7, 2020.
- [35] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation: a unifying framework for information theoretic feature selection. *The journal of machine learning research*, 13(1):27–66, 2012.
- [36] Alexander E. I. Brownlee, Olivier Regnier-Coudert, John A. W. McCall, and Stewart Massie. Using a markov network as a surrogate fitness function in a genetic algorithm. In *IEEE Congress on Evolutionary Computation*, pages 1–8, New York City, USA, 2010. IEEE. doi: 10.1109/CEC.2010.5586548.
- [37] Alexander E.I. Brownlee. Mining markov network surrogates for value-added optimisation. In *Proceedings of GECCO 2016*, GECCO '16 Companion, pages 1267–1274, New York, NY, USA, 2016. ACM. ISBN 9781450343237. doi: 10.1145/2908961.2931711.
- [38] Alexander EI Brownlee, John AW McCall, and Lee A Christie. Structural coherence of problem and algorithm: An analysis for edas on all 2-bit and 3-bit problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2066–2073. IEEE, 2015.
- [39] Nick Bryan-Kinns, Corey Ford, Alan Chamberlain, Steven David Benford, Helen Kennedy, Zijin Li, Wu Qiong, Gus G Xia, and Jeba Rezwana. Explainable ai for the arts: Xaixarts. In *Proceedings of the 15th Conference on Creativity and Cognition*, pages 1–7, 2023.
- [40] Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.
- [41] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-based clustering based on hierarchical density estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-37456-2.
- [42] GianCarlo A. P. I. Catalano, Alexander E. I. Brownlee, David Cairns, John McCall, and Russell Ainslie. Evaluating a novel explainability method for metaheuristics via a user study. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '25 Companion, pages 1984–1991, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714641. doi: 10.1145/3712255.3734337. URL <https://doi.org/10.1145/3712255.3734337>.
- [43] GianCarlo A. P. I. Catalano, Klaudia Dynak, Alexander E. I. Brownlee, and Piotr Lipinski. Explaining recommender systems' performance via user behaviour patterns. In Max Bramer and Frederic Stahl, editors, *Artificial Intelligence XLII*, pages 35–47, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-11442-6. doi: 10.1007/978-3-032-11442-6\_3.
- [44] Giancarlo Antonino Pasquale Ignazio Catalano, Alexander E. I Brownlee, David Cairns, John Mccall, and Russell Ainslie. Mining potentially explanatory patterns via partial solutions. GECCO '24 Companion, pages 567–570, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704956. doi: 10.1145/3638530.3654318. URL <https://doi.org/10.1145/3638530.3654318>.

- [45] GianCarlo Antonino Pasquale Ignazio Catalano, Alexander Brownlee, David Cairns, Russell Ainslie, and John McCall. Interpretable decision trees to predict solution fitness. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '25*, pages 1099–1107, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400714658. doi: 10.1145/3712256.3726387. URL <https://doi.org/10.1145/3712256.3726387>.
- [46] Giancarlo API Catalano, Alexander EI Brownlee, David Cairns, John AW McCall, Martin Fyvie, and Russell Ainslie. Explaining a staff rostering problem using partial solutions. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 179–193. Springer, 2024. doi: [https://doi.org/10.1007/978-3-031-77918-3\\_13](https://doi.org/10.1007/978-3-031-77918-3_13).
- [47] Suyog Chandramouli, Yifan Zhu, and Antti Oulasvirta. Interactive personalization of classifiers for explainability using multi-objective bayesian optimization. In *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization, UMAP '23*, pages 34–45, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450399326. doi: 10.1145/3565472.3592956. URL <https://doi.org/10.1145/3565472.3592956>.
- [48] Sourav Chatterjee. A survey of some recent developments in measures of association. *Probability and stochastic processes: a volume in Honour of Rajeeva L. Karandikar*, pages 109–128, 2024.
- [49] Stephen Chen, James Montgomery, and Antonio Bolufé-Röhler. Some measurements on the effects of the curse of dimensionality. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, GECCO Comp '14*, pages 1447–1448, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450328814. doi: 10.1145/2598394.2602271. URL <https://doi.org/10.1145/2598394.2602271>.
- [50] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.
- [51] W. Chen, D. Ghosh, T.E. Raghunathan, M. Norkin, D.J. Sargent, and G. Bepler. On bayesian methods of exploring qualitative interactions for targeted treatment. *Statistics in Medicine*, 31(28):3693–3707, 2012.
- [52] Chung-Kuan Cheng, Andrew B. Kahng, Sayak Kundu, Yucheng Wang, and Zhiang Wang. An updated assessment of reinforcement learning for macro placement, 2026. URL <https://arxiv.org/abs/2302.11014>.
- [53] Shiwei Cheng and Anind K Dey. I see, you design: user interface intelligent design system with eye tracking and interactive genetic algorithm. *CCF Transactions on Pervasive Computing and Interaction*, 1(3):224–236, 2019.
- [54] Yao-Hsin Chou, Cheng-Yen Hua, Shu-Yu Kuo, Yu-Chi Jiang, and Sy-Yen Kuo. Evo-panel: Dynamic visualization tool for optimization process. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 29–30, 2024.
- [55] Michael Chromik, Malin Eiband, Felicitas Buchner, Adrian Krüger, and Andreas Butz. I think i get your point, ai! the illusion of explanatory depth in explainable ai. In *Proceedings of the 26th International Conference on Intelligent User Interfaces, IUI '21*, pages 307–317, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380171. doi: 10.1145/3397481.3450644. URL <https://doi.org/10.1145/3397481.3450644>.
- [56] Kristi Coale. Darwin in a box, July 14 1997. URL <https://www.wired.com/1997/07/darwin-in-a-box/>. Accessed: 2025-10-07.
- [57] Carlos A Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, 191(11-12):1245–1287, 2002.
- [58] Carlos A Coello Coello, Gary B Lamont, and David A Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.

- [59] David Coffin and Robert E. Smith. *Linkage Learning in Estimation of Distribution Algorithms*, pages 141–156. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-85068-7. doi: 10.1007/978-3-540-85068-7\_7. URL [https://doi.org/10.1007/978-3-540-85068-7\\_7](https://doi.org/10.1007/978-3-540-85068-7_7).
- [60] David Jonathan Coffin and Robert Elliott Smith. The limitations of distribution sampling for linkage learning. In *2007 IEEE Congress on Evolutionary Computation*, pages 364–369. IEEE, 2007.
- [61] David A. Cohen, Martin C. Cooper, Peter G. Jeavons, and Andrei A. Krokhin. The complexity of soft constraint satisfaction. *Artificial Intelligence*, 170(11):983–1016, 2006. ISSN 0004-3702. doi: <https://doi.org/10.1016/j.artint.2006.04.002>. URL <https://www.sciencedirect.com/science/article/pii/S000437020600052X>.
- [62] Roberto Confalonieri and Jose Maria Alonso-Moral. An operational framework for guiding human evaluation in explainable and trustworthy artificial intelligence. *IEEE Intelligent Systems*, 39(1):18–28, 2024. doi: 10.1109/MIS.2023.3334639.
- [63] Matt Cooper. Anova - type i/ii/iii ss explained, 2026. URL [https://md.psych.bio.uni-goettingen.de/mv/unit/lm\\_cat/lm\\_cat\\_unbal\\_ss\\_explained.html](https://md.psych.bio.uni-goettingen.de/mv/unit/lm_cat/lm_cat_unbal_ss_explained.html). Accessed: 2026-03-10.
- [64] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.
- [65] H. Cramer. *Mathematical Methods of Statistics*. Princeton University Press, Princeton, 1946. Second paragraph.
- [66] Noel Cressie and Timothy R. C. Read. Multinomial goodness-of-fit tests. *Journal of the Royal Statistical Society. Series B. Methodological*, 46(3):440–464, 1984. Retrieved 14 January 2026.
- [67] Marcin Czajkowski, Krzysztof Jurczuk, and Marek Kretowski. Steering the interpretability of decision trees using lasso regression-an evolutionary perspective. *Information Sciences*, 638:118944, 2023.
- [68] Marina Danilova, Pavel Dvurechensky, Alexander Gasnikov, Eduard Gorbunov, Sergey Guminov, Dmitry Kamzolov, and Innokentiy Shibaev. Recent theoretical advances in non-convex optimization. In *High-Dimensional Optimization and Probability: With a View Towards Data Science*, pages 79–163. Springer, 2022.
- [69] Indraneel Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998. doi: 10.1137/S1052623496307510. URL <https://doi.org/10.1137/S1052623496307510>.
- [70] Jeremy De Bonet, Charles Isbell, and Paul Viola. Mimic: Finding optima by estimating probability densities. *Advances in neural information processing systems*, 9, 1996.
- [71] B Bart de Langhe, Stefano Puntoni, and Richard P Larrick. Linear thinking in a nonlinear world. *Harvard Business Review*, 95(3):130–139, 2017.
- [72] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.
- [73] Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pages 3–34. Springer, 2011.
- [74] Kalyanmoy Deb and Tushar Goel. Controlled elitist non-dominated sorting genetic algorithms for better convergence. In *International conference on evolutionary multi-criterion optimization*, pages 67–81. Springer, 2001.
- [75] Kalyanmoy Deb and David E Goldberg. Sufficient conditions for deceptive and easy binary functions. *Annals of mathematics and Artificial Intelligence*, 10(4): 385–408, 1994.

- [76] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints. *IEEE transactions on evolutionary computation*, 18(4):577–601, 2013.
- [77] Kalyanmoy Deb and Aravind Srinivasan. Innovization: Discovery of innovative design principles through multiobjective evolutionary optimization. *Multiobjective Problem Solving from Nature*, pages 243–262, 2008.
- [78] Kalyanmoy Deb and Santosh Tiwari. Omni-optimizer: A procedure for single and multi-objective optimization. In *International conference on evolutionary multi-criterion optimization*, pages 47–61. Springer, 2005.
- [79] Persi Diaconis and R. L. Graham. Spearman’s footrule as a measure of disarray. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(2):262–268, 12 2018. ISSN 0035-9246. doi: 10.1111/j.2517-6161.1977.tb01624.x. URL <https://doi.org/10.1111/j.2517-6161.1977.tb01624.x>.
- [80] Boty Dimanov, Umang Bhatt, Mateja Jamnik, and Adrian Weller. You shouldn’t trust me: Learning models which conceal unfairness from multiple explanation methods. In *ECAI 2020*, pages 2473–2480. IOS Press, 2020.
- [81] Mary Dimitropoulaki, Mathias Kern, Gilbert Owusu, and Alistair McCormick. Workforce rostering via metaheuristics. In Max Bramer and Miltos Petridis, editors, *Artificial Intelligence XXXV*, pages 277–290, Cham, 2018. Springer International Publishing. ISBN 978-3-030-04191-5.
- [82] Weiping Ding, Mohamed Abdel-Basset, Hossam Hawash, and Ahmed M. Ali. Explainability of artificial intelligence methods, applications and challenges: A comprehensive survey. *Information Sciences*, 615:238–292, 2022. ISSN 0020-0255. doi: <https://doi.org/10.1016/j.ins.2022.10.013>. URL <https://www.sciencedirect.com/science/article/pii/S002002552201132X>.
- [83] Finale Doshi-Velez and Been Kim. Considerations for evaluation and generalization in interpretable machine learning. *Explainable and interpretable models in computer vision and machine learning*, pages 3–17, 2018.
- [84] Gunning D Aha DW. Darpa’s explainable artificial intelligence program. *AI Mag*, 40(2):44, 2019.
- [85] Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, and Rajiv Ranjan. Explainable ai (xai): Core ideas, techniques, and solutions. *ACM Comput. Surv.*, 55(9), January 2023. ISSN 0360-0300. doi: 10.1145/3561048. URL <https://doi.org/10.1145/3561048>.
- [86] Tome Eftimov, Gorjan Popovski, Dragi Kocev, and Peter Korošec. Performance2vec: a step further in explainable stochastic optimization algorithm performance. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, pages 193–194, 2020.
- [87] Upol Ehsan and Mark O Riedl. Explainability pitfalls: Beyond dark patterns in explainable ai. *Patterns*, 5(6), 2024.
- [88] Emerging Technology from the arXiv. Evolutionary algorithm outperforms deep-learning machines at video games, July 2018. URL <https://www.technologyreview.com/2018/07/18/104191/evolutionary-algorithm-outperforms-deep-learning-machines-at-video-games/>.
- [89] Floriana Esposito, Donato Malerba, Giovanni Semeraro, and J Kay. A comparative analysis of methods for pruning decision trees. *IEEE transactions on pattern analysis and machine intelligence*, 19(5):476–491, 1997.
- [90] David J Finney. Main effects and interactions. *Journal of the American Statistical Association*, 43(244):566–571, 1948.
- [91] Jack McKay Fletcher and Thomas Wennekers. A natural approach to studying schema processing, 2017.

- [92] Jane Forman and Laura Damschroder. *Qualitative content analysis*. Elsevier, 2007.
- [93] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553, September 1983. doi: 10.2307/2288117.
- [94] Timo Freiesleben and Gunnar König. Dear xai community, we need to talk! fundamental misconceptions in current xai research. In *World Conference on Explainable Artificial Intelligence*, pages 48–65. Springer, 2023.
- [95] Jerome H. Friedman and Bogdan E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3), September 2008. ISSN 1932-6157. doi: 10.1214/07-aos148. URL <http://dx.doi.org/10.1214/07-AOS148>.
- [96] Martin Fyvie. *Explainability of Non-Deterministic Solvers: Explanatory Feature Generation from the Data Mining of the Search Trajectories of Population-Based Metaheuristics*. PhD thesis, Robert Gordon University, 2024.
- [97] Martin Fyvie, John A. W. McCall, Lee A. Christie, Alexander E. I. Brownlee, and Manjinder Singh. Towards explainable metaheuristics: Feature extraction from trajectory mining. *Expert Systems*, page e13494, 2021. doi: 10.1111/exsy.13494.
- [98] Martin Fyvie, John AW McCall, and Lee A Christie. Non-deterministic solvers and explainable ai through trajectory mining. In *CEUR*. CEUR Workshop Proceedings, 2021.
- [99] Martin Fyvie, John A.W. McCall, and Lee A. Christie. Non-deterministic solvers and explainable ai through trajectory mining. *SICSA XAI Workshop 2021*, 2894: 75–78, 6 2021. ISSN 1613-0073. URL <https://rgu-repository.worktribe.com/output/1395881>.
- [100] Martin Fyvie, John Mccall, Lee Christie, and Alexander Brownlee. Explaining a staff rostering genetic algorithm using sensitivity analysis and trajectory analysis. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 1648–1656, 2023.
- [101] Martin Fyvie, John Mccall, Lee Christie, and Alexander Brownlee. Explaining a staff rostering genetic algorithm using sensitivity analysis and trajectory analysis. In *Proceedings of GECCO '13*, pages 1648–1656, 2023.
- [102] Martin Fyvie, John AW McCall, Lee A Christie, Alexander EI Brownlee, and Manjinder Singh. Towards explainable metaheuristics: Feature extraction from trajectory mining. *Expert Systems*, page e13494, 2023.
- [103] Gautam Garai. Application of genetic algorithm in numerous scientific fields. In *Genetic algorithms*. IntechOpen, 2022.
- [104] Abhinav Gaur and Kalyanmoy Deb. Towards an automated innovization method for handling discrete search spaces. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2899–2906. IEEE, 2015.
- [105] Ali Ghaheri, Saeed Shoar, Mohammad Naderan, and Sayed shahabuddin Hoseini. The applications of genetic algorithms in medicine. *Oman Medical Journal*, 30: 406–416, 11 2015. doi: 10.5001/omj.2015.82.
- [106] NN Glibovets and Nataliya M Gulayeva. A review of niching genetic algorithms for multimodal function optimization. *Cybernetics and Systems Analysis*, 49(6): 815–820, 2013.
- [107] David E Goldberg. Genetic algorithms and walsh functions: Part 2, deception and its analysis. *Complex systems*, 3:153–171, 1989.
- [108] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.

- [109] David E. Goldberg, Kumara Sastry, and Yukio Ohsawa. *Discovering Deep Building Blocks for Competent Genetic Algorithms Using Chance Discovery via KeyGraphs*, pages 276–301. Springer, Berlin, Heidelberg, Tokyo, Japan, 2003. ISBN 978-3-662-06230-2. doi: 10.1007/978-3-662-06230-2\_19.
- [110] David E. Gomes, Maria In  s D. Igl  sias, Ana P. Proen  a, T  nia M. Lima, and Pedro D. Gaspar. Applying a genetic algorithm to a m-tsp: Case study of a decision support system for optimizing a beverage logistics vehicles routing problem. *Electronics*, 10(18), 2021. ISSN 2079-9292. doi: 10.3390/electronics10182298.
- [111] Cheng Gong, Yang Nan, Meng Pang, Hisao Ishibuchi, and Qingfu Zhang. Performance of nsga-iii on multi-objective combinatorial optimization problems heavily depends on its implementations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 511–519, 2024.
- [112] Alicja Gosiewska and Przemyslaw Biecek. Do not trust additive explanations. *arXiv preprint arXiv:1903.11420*, 2019.
- [113] Peter Grunwald and Paul Vitanyi. Shannon information and kolmogorov complexity, 2004. URL <https://arxiv.org/abs/cs/0410002>.
- [114] David Gunning, Eric Vorm, Yunyan Wang, and Matt Turek. Darpa’s explainable ai (xai) program: A retrospective. *Authorea Preprints*, 2021.
- [115] Mark A Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.
- [116] Jonathan Hallam and A Prugel-Bennett. Large barrier trees for studying search. *IEEE Transactions on Evolutionary Computation*, 9(4):385–397, 2005.
- [117] Georges R Harik and David E Goldberg. Learning linkage. In *FOGA*, volume 4, pages 247–262, 1996.
- [118] Georges R Harik, Fernando G Lobo, and Kumara Sastry. Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ecga). In *Scalable optimization via probabilistic modeling*, pages 39–61. Springer, 2006.
- [119] Mark Harman. Search based software engineering. In *International Conference on Computational Science*, pages 740–747. Springer, 2006.
- [120] Ahmad Hassanat, Khalid Almohammadi, Esra’a Alkafaween, Eman Abunawas, Awni Hammouri, and VB Surya Prasath. Choosing mutation and crossover ratios for genetic algorithms-a review with a new dynamic approach. *Information*, 10(12):390, 2019.
- [121] Vikas Hassija, Vinay Chamola, Atmesh Mahapatra, Abhinandan Singal, Divyansh Goel, Kaizhu Huang, Simone Scardapane, Indro Spinelli, Mufti Mahmud, and Amir Hussain. Interpreting black-box models: a review on explainable artificial intelligence. *Cognitive Computation*, 16(1):45–74, 2024.
- [122] Trevor J Hastie. Generalized additive models. *Statistical models in S*, pages 249–307, 2017.
- [123] Mark Hauschild and Martin Pelikan. An introduction and survey of estimation of distribution algorithms. *Swarm and evolutionary computation*, 1(3):111–128, 2011.
- [124] Cheng He, Ye Tian, and Zhichao Lu. Artificial evolutionary intelligence (AEI): Evolutionary computation evolves with large language models. *Journal of Membrane Computing*, 7(2):135–152, June 2025. ISSN 2523-8914. doi: 10.1007/s41965-024-00172-x. URL <https://doi.org/10.1007/s41965-024-00172-x>.
- [125] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648, 2020.

- [126] David Heath, Simon Kasif, and Steven Salzberg. Learning oblique decision trees. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1002–1007. Morgan Kaufmann Burlington, MA, 1993.
- [127] Robert B. Heckendorn and Alden H. Wright. Efficient Linkage Discovery by Limited Probing. *Evolutionary Computation*, 12(4):517–545, 12 2004. ISSN 1063-6560. doi: 10.1162/1063656043138914.
- [128] Robert Baker Heckendorn. *Walsh analysis, epistasis, and optimization problem difficulty for evolutionary algorithms*. Colorado State University, 1999.
- [129] Lukas-Valentin Herm. Impact of explainable ai on cognitive load: Insights from an empirical study. *arXiv preprint arXiv:2304.08861*, 2023.
- [130] Lukas-Valentin Herm, Kai Heinrich, Jonas Wanner, and Christian Janiesch. Stop ordering machine learning algorithms by their explainability! a user-centered investigation of performance and explainability. *International Journal of Information Management*, 69:102538, 2023.
- [131] David G. Herr. On the history of anova in unbalanced, factorial designs: The first 30 years. *The American Statistician*, 40(4):265–270, 1986. ISSN 00031305. URL <http://www.jstor.org/stable/2684597>.
- [132] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. University of Michigan Press, 1975.
- [133] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, Massachusetts, USA, 04 1992. ISBN 9780262275552. doi: 10.7551/mitpress/1090.001.0001.
- [134] Andreas Holzinger, Anna Saranti, Christoph Molnar, Przemyslaw Biecek, and Wojciech Samek. Explainable ai methods-a brief overview. In *International workshop on extending explainable AI beyond deep models and classifiers*, pages 13–38. Springer, 2022.
- [135] Holger H Hoos and Thomas Stützle. Satlib: An online resource for research on sat. *Sat*, 2000:283–292, 2000.
- [136] Shih-Huan Hsu and Tian-Li Yu. Optimization by pairwise linkage detection, incremental linkage set, and restricted/back mixing: Dsmga-ii. In *Proceedings of GECCO 2015*, pages 519–526, New York, NY, USA, 2015. ACM.
- [137] Lingying Huang, Xiaomeng Chen, Wei Huo, Jiazheng Wang, Fan Zhang, Bo Bai, and Ling Shi. Branch and bound in mixed integer linear programming problems: A survey of techniques and trends, 2021. URL <https://arxiv.org/abs/2111.06257>.
- [138] Wei Huang, Xingyu Zhao, Gaojie Jin, and Xiaowei Huang. Safari: Versatile and efficient evaluations for robustness of interpretability. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1988–1998, 2023.
- [139] Kelly Hunter, Sarah L Thomson, and Emma Hart. Into the black box: mining variable importance with xai. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 20–35. Springer, 2025.
- [140] Kelly Hunter, Sarah L Thomson, and Emma Hart. Variable importance estimation for high-dimensional optimisation. In *UK Workshop on Computational Intelligence*, pages 115–126. Springer, 2025.
- [141] Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. Performance comparison of nsga-ii and nsga-iii on various many-objective test problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3045–3052. IEEE, 2016.
- [142] Yacine Izza, Alexey Ignatiev, and Joao Marques-Silva. On explaining decision trees. *arXiv preprint arXiv:2010.11034*, 2020.

- [143] Hannu Jaakkola, Jaak Henno, Jukka Mäkelä, and Bernhard Thalheim. Artificial intelligence yesterday, today and tomorrow. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 860–867. IEEE, 2019.
- [144] James Jaccard and Kim Daniloski. Analysis of variance and the general linear model. 2012.
- [145] Alon Jacovi, Ana Marasović, Tim Miller, and Yoav Goldberg. Formalizing trust in artificial intelligence: Prerequisites, causes and goals of human trust in ai. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 624–635, 2021.
- [146] Tommy R Jensen and Bjarne Toft. *Graph coloring problems*. John Wiley & Sons, 2011.
- [147] Chenxia Jin, Fachao Li, Marzana Wilamowska-Korsak, Ling Li, and Liuliu Fu. Bsp-ga: A new genetic algorithm for system optimization and excellent schema selection. *Systems Research and Behavioral Science*, 31:337–352, 5 2014. ISSN 1099-1743. doi: 10.1002/SRES.2280.
- [148] ZQ John Lu. The elements of statistical learning: data mining, inference, and prediction, 2010.
- [149] Terry Jones et al. *Evolutionary algorithms, fitness landscapes and search*. PhD thesis, University of New Mexico Albuquerque, NM, 1995.
- [150] Kenneth A. De Jong and William M. Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 5(1):1–26, 1992. ISSN 1573-7470. doi: 10.1007/BF01530777. URL <https://doi.org/10.1007/BF01530777>.
- [151] Thomas Joyce and J. Michael Herrmann. *A Review of No Free Lunch Theorems, and Their Implications for Metaheuristic Optimisation*, pages 27–51. Springer International Publishing, Cham, 2018. ISBN 978-3-319-67669-2. doi: 10.1007/978-3-319-67669-2\_2. URL [https://doi.org/10.1007/978-3-319-67669-2\\_2](https://doi.org/10.1007/978-3-319-67669-2_2).
- [152] Bryant A. Julstrom. Greedy, genetic, and greedy genetic algorithms for the quadratic knapsack problem. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, GECCO '05*, pages 607–614, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595930108. doi: 10.1145/1068009.1068111. URL <https://doi.org/10.1145/1068009.1068111>.
- [153] Josef Kallrath and Anna Schrieck. Discrete optimisation and real world problems. In Bob Hertzberger and Giuseppe Serazzi, editors, *High-Performance Computing and Networking*, pages 351–359, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. ISBN 978-3-540-49242-9.
- [154] Richard M Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*, pages 219–241. Springer, 2009.
- [155] Stuart A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, New York, 1993. URL <https://ia601606.us.archive.org/30/items/stuart-a-kauffman-the-origins-of-order-selforganization-and-complexity-in-evolution/stuart-a-kauffman-the-origins-of-order-selforganization-and-complexity-in-evolution.pdf>. PDF available from Internet Archive.
- [156] Pascal Kerschke and Heike Trautmann. Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the r-package flacco. In *Applications in statistical computing: from music data analysis to industrial quality improvement*, pages 93–123. Springer, 2019.
- [157] Samina Khalid, Tehmina Khalil, and Shamila Nasreen. A survey of feature selection and feature extraction techniques in machine learning. In *2014 science and information conference*, pages 372–378. IEEE, 2014.

- [158] Asifullah Khan, Aqsa Saeed Qureshi, Noorul Wahab, Mutawarra Hussain, and Muhammad Yousaf Hamza. A recent survey on the applications of genetic programming in image processing. *Computational Intelligence*, 37(4):1745–1778, 2021.
- [159] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016.
- [160] Hae-Young Kim. Statistical notes for clinical researchers: Two-way analysis of variance (anova)-exploring possible interaction between factors. *Restorative dentistry & endodontics*, 39(2):143–147, 2014.
- [161] Hae-Young Kim. Statistical notes for clinical researchers: Chi-squared test and fisher’s exact test. *Restorative Dentistry and Endodontics*, 42:152–155, 05 2017. doi: 10.5395/rde.2017.42.2.152.
- [162] Sotiris B Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39:261–283, 2013.
- [163] Mark W Krentel. The complexity of optimization problems. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 69–76, 1986.
- [164] Satyapriya Krishna, Tessa Han, Alex Gu, Steven Wu, Shahin Jabbari, and Himabindu Lakkaraju. The disagreement problem in explainable machine learning: A practitioner’s perspective. *arXiv preprint arXiv:2202.01602*, 2022.
- [165] Jan Kruse, Andy M Connor, and Stefan Marks. Evaluation of a multi-agent "human-in-the-loop" game design system. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 12(3):1–26, 2022.
- [166] Sushant Kumar, Sumit Datta, Vishakha Singh, Deepanwita Datta, Sanjay Kumar Singh, and Ritesh Sharma. Applications, challenges, and future directions of human-in-the-loop learning. *IEEE Access*, 12:75735–75760, 2024.
- [167] Nikos D Lagaros, Makis Kournoutos, Nikos Ath Kallioras, and Alexandros N Nordas. Constraint handling techniques for metaheuristics: a state-of-the-art review and new variants. *Optimization and Engineering*, 24(4):2251–2298, 2023.
- [168] Himabindu Lakkaraju and Osbert Bastani. "how do i fool you?": Manipulating user trust via misleading black box explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, AIES ’20*, pages 79–85, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371100. doi: 10.1145/3375627.3375833. URL <https://doi.org/10.1145/3375627.3375833>.
- [169] Øyvind Langsrud. Anova for unbalanced data: Use type ii instead of type iii sums of squares. *Statistics and computing*, 13(2):163–167, 2003.
- [170] Pedro Larrañaga and Jose A Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer Science & Business Media, 2001.
- [171] Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.
- [172] Yuri Lavinias, Claus Aranha, Tetsuya Sakurai, and Marcelo Ladeira. Experimental analysis of the tournament size on genetic algorithms. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3647–3653, 2018. doi: 10.1109/SMC.2018.00617.
- [173] Fernando Lezama, Jose Almeida, Joao Soares, and Zita Vale. Explainergy: Towards explainability of metaheuristic performance in the energy field. In *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 783–788. IEEE, 2023.
- [174] Junbing Li, Changqing Zhang, Joey Tianyi Zhou, Huazhu Fu, Shuyin Xia, and Qinghua Hu. Deep-lift: Deep label-specific feature learning for image annotation. *IEEE transactions on Cybernetics*, 52(8):7732–7741, 2021.

- [175] Qiqiang LI and Yongchao GAO. Backbone based extremal evolutionary optimization for combinatorial problems.
- [176] Xiaodong Li, Michael G Epitropakis, Kalyanmoy Deb, and Andries Engelbrecht. Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation*, 21(4):518–538, 2016.
- [177] Yi-Chi Liao. Human-in-the-loop design optimization, 2023.
- [178] Zachary C. Lipton. The mythos of model interpretability, 2017. URL <https://arxiv.org/abs/1606.03490>.
- [179] Dazhuang Liu, Marco Virgolin, Tanja Alderliesten, and Peter AN Bosman. Evolvability degeneration in multi-objective genetic programming for symbolic regression. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 973–981, 2022.
- [180] Jie Liu. *Effective User Interfaces for Human-in-the-loop Optimisation*. PhD thesis, Monash University, 2019.
- [181] Xavier Llorca and Josep M Garrell. Evolution of decision trees. In *Proceedings of the 4th Catalan Conference on Artificial Intelligence*, pages 115–122. Citeseer, 2001.
- [182] Luca Longo, Randy Goebel, Freddy Lecue, Peter Kieseberg, and Andreas Holzinger. Explainable artificial intelligence: Concepts, applications, research challenges and visions. In Andreas Holzinger, Peter Kieseberg, A Min Tjoa, and Edgar Weippl, editors, *Machine Learning and Knowledge Extraction*, pages 1–16, Cham, 2020. Springer International Publishing. ISBN 978-3-030-57321-8.
- [183] Luca Longo, Mario Brcic, Federico Cabitza, Jaesik Choi, Roberto Confalonieri, Javier Del Ser, Riccardo Guidotti, Yoichi Hayashi, Francisco Herrera, Andreas Holzinger, et al. Explainable artificial intelligence (xai) 2.0: A manifesto of open challenges and interdisciplinary research directions. *Information Fusion*, 106: 102301, 2024.
- [184] Ana Lucic, Hinda Haned, and Maarten de Rijke. Why does my model fail? contrastive local explanations for retail forecasting. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 90–98, 2020.
- [185] Sean Luke. *Essentials of Metaheuristics*. Lulu, second edition, 2013. Available for free at <http://cs.gmu.edu/~sean/book/metaheuristics/>.
- [186] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.
- [187] R Timothy Marler and Jasbir S Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and multidisciplinary optimization*, 41(6):853–862, 2010.
- [188] Elio Masciari, Areeba Umair, and Muhammad Habib Ullah. A systematic literature review on ai-based recommendation systems and their ethical considerations. *IEEE Access*, 12:121223–121241, 2024.
- [189] J.H. McDonald. G-test of goodness-of-fit. In *Handbook of Biological Statistics*, pages 53–58. Sparky House Publishing, Baltimore, Maryland, third edition, 2014.
- [190] Yi Mei, Qi Chen, Andrew Lensen, Bing Xue, and Mengjie Zhang. Explainable artificial intelligence by genetic programming: A survey. *IEEE Transactions on Evolutionary Computation*, 27(3):621–641, 2022.
- [191] Merriam-Webster, Inc. Explaining definition and meaning, n.d. URL <https://www.merriam-webster.com/dictionary/explaining>. Accessed: 2026-02-21.
- [192] Michael Merry, Pat Riddle, and Jim Warren. A mental models approach for defining explainable artificial intelligence. *BMC Medical Informatics and Decision Making*, 21(1):344, 2021.

- [193] Ljubiša Metikoš, Dr Clara Iglesias Keller, Cong-rui Qiao, and Natali Helberger. Comparing the right to an explanation of judicial ai by function: Studies on the eu, brazil, and china. *Brazil, and China (October 28, 2025)*, 2025.
- [194] George A Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.
- [195] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.
- [196] Dang Minh, H Xiang Wang, Y Fen Li, and Tan N Nguyen. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, pages 1–66, 2022.
- [197] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nova, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862): 207–212, 2021.
- [198] Melanie Mitchell, Stephanie Forrest, and John Holland. The royal road for genetic algorithms: Fitness landscapes. *European Conference on Artificial Life*, 1:23–33, 11 1992.
- [199] Melanie Mitchell, John Holland, and Stephanie Forrest. When will a genetic algorithm outperform hill climbing. *Advances in neural information processing systems*, 6, 1993.
- [200] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-wise relevance propagation: an overview. *Explainable AI: interpreting, explaining and visualizing deep learning*, pages 193–209, 2019.
- [201] David E Moriarty, Alan C Schultz, and John J Grefenstette. Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:241–276, 1999.
- [202] Masaharu Munetomo and David Goldberg. Identifying linkage by nonlinearity check. *IlIiGAi report n. 98012*, 11 1998.
- [203] Carsten Murawski and Peter Bossaerts. How humans solve complex problems: The case of the knapsack problem. *Scientific reports*, 6(1):34851, 2016.
- [204] Giorgia Nadizar, Luigi Rovito, Andrea De Lorenzo, Eric Medvet, and Marco Virgolin. An analysis of the ingredients for learning interpretable symbolic regression models with human-in-the-loop and genetic programming. *ACM Transactions on Evolutionary Learning and Optimization*, 4(1):1–30, 2024.
- [205] Luca Nannini. Habemus a right to an explanation: so what?—a framework on transparency-explainability functionality and tensions in the eu ai act. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, volume 7, pages 1023–1035, 2024.
- [206] Luca Nannini. Xai: regulatory and ethical aspects. In *Explainable AI for Communications and Networking*, pages 181–207. Elsevier, 2025.
- [207] Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmin Schmitt, Jörg Schlötterer, Maurice van Keulen, and Christin Seifert. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. *ACM Comput. Surv.*, 55(13s), July 2023. ISSN 0360-0300. doi: 10.1145/3583558. URL <https://doi.org/10.1145/3583558>.
- [208] Gabriela Ochoa. Setting the mutation rate: Scope and limitations of the 1/l heuristic. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 495–502, 2002.
- [209] Gabriela Ochoa, Sébastien Verel, Fabio Daolio, and Marco Tomassini. Local optima networks: A new model of combinatorial fitness landscapes. *Recent advances in the theory and application of fitness landscapes*, pages 233–262, 2014.

- [210] Gabriela Ochoa, Katherine M Malan, and Christian Blum. Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics. *Applied Soft Computing*, 109:107492, 2021.
- [211] Mohammad Nabi Omidvar, Xiaodong Li, and Xin Yao. A review of population-based metaheuristics for large-scale black-box global optimization-part i. *IEEE Transactions on Evolutionary Computation*, 26(5):802–822, 2021.
- [212] Andrés Páez. Axe the x in xai: A plea for understandable ai. In *Philosophy of science for machine learning: Core issues and new perspectives*, pages 133–156. Springer, 2025.
- [213] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, New York, 2 edition, 1984.
- [214] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. URL <http://jmlr.org/papers/v12/pedregosa11a.html>.
- [215] Martin Pelikan. *Probabilistic Model-Building Genetic Algorithms*, pages 13–30. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-32373-0. doi: 10.1007/978-3-540-32373-0\_2. URL [https://doi.org/10.1007/978-3-540-32373-0\\_2](https://doi.org/10.1007/978-3-540-32373-0_2).
- [216] Martin Pelikan. Hierarchical bayesian optimization algorithm. In *Hierarchical Bayesian Optimization Algorithm: Toward a new Generation of Evolutionary Algorithms*, pages 105–129. Springer, 2005.
- [217] Martin Pelikan. Nk landscapes, problem difficulty, and hybrid evolutionary algorithms. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 665–672, 2010.
- [218] Martin Pelikan and David E Goldberg. Escaping hierarchical traps with competent genetic algorithms. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 511–518, 2001.
- [219] Martin Pelikan and Heinz Muehlenbein. The bivariate marginal distribution algorithm. In Rajkumar Roy, Takeshi Furuhashi, and Pravir K. Chawdhry, editors, *Advances in Soft Computing*, pages 521–535, London, 1999. Springer London. ISBN 978-1-4471-0819-1.
- [220] Fernando Peres and Mauro Castelli. Combinatorial optimization problems and metaheuristics: Review, challenges, design, and development. *Applied Sciences*, 11(14), 2021. ISSN 2076-3417. doi: 10.3390/app11146449. URL <https://www.mdpi.com/2076-3417/11/14/6449>.
- [221] P. Jonathon Phillips, Carina Hahn, Peter Fontana, Amy Yates, Kristen K. Greene, David Broniatowski, and Mark A. Przybocki. Four principles of explainable artificial intelligence, 2021-09-29 04:09:00 2021.
- [222] Wang Ping. Application of genetic algorithm in logistics management and distribution. In *International Conference on Multi-modal Information Analytics*, pages 1041–1046. Springer, 2022.
- [223] Richard J Preen, Larry Bull, and Andrew Adamatzky. Towards an evolvable cancer treatment simulator. *Biosystems*, 182:1–7, 2019.
- [224] J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.
- [225] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.
- [226] Denise Rey and Markus Neuhäuser. *Wilcoxon-Signed-Rank Test*, pages 1658–1659. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-04898-2. doi: 10.1007/978-3-642-04898-2\_616. URL [https://doi.org/10.1007/978-3-642-04898-2\\_616](https://doi.org/10.1007/978-3-642-04898-2_616).

- [227] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA, 2016. ACM. ISBN 9781450342322. doi: 10.1145/2939672.2939778.
- [228] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [229] John Michael Robson. A new proof of the NP-completeness of satisfiability. In *Proceedings of the 2nd Australian Computer Science Conference*, pages 62–70, February 1979.
- [230] Olympia Roeva. *Real-world applications of genetic algorithms*. Books on Demand, 2012.
- [231] Yao Rong, Tobias Leemann, Thai-Trang Nguyen, Lisa Fiedler, Peizhu Qian, Vaibhav Unhelkar, Tina Seidel, Gjergji Kasneci, and Enkelejda Kasneci. Towards human-centered explainable ai: A survey of user studies for model explanations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(4): 2104–2122, 2024. doi: 10.1109/TPAMI.2023.3331846.
- [232] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5): 206–215, 2019.
- [233] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges, 2021. URL <https://arxiv.org/abs/2103.11251>.
- [234] Ahmed M Salih and Yuhe Wang. Are linear regression models white box and interpretable? *arXiv preprint arXiv:2407.12177*, 2024.
- [235] Ahmed M. Salih, Zahra Raisi-Estabragh, Ilaria Boscolo Galazzo, Petia Radeva, Steffen E. Petersen, Karim Lekadir, and Gloria Menegaz. A perspective on explainable artificial intelligence methods: Shap and lime. *Advanced Intelligent Systems*, 7(1):2400304, 2025. doi: <https://doi.org/10.1002/aisy.202400304>. URL <https://advanced.onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202400304>.
- [236] Leonard J Savage. The theory of statistical decision. *Journal of the American Statistical association*, 46(253):55–67, 1951.
- [237] Thomas Schläpfer, David Dunning, Kerri L. Johnson, and Justin Kruger. How unaware are the unskilled? empirical tests of the signal extraction counter-explanation for the dunning-kruger effect in self-evaluation of performance. *Journal of Economic Psychology*, 39:85–100, 2013. ISSN 0167-4870. doi: <https://doi.org/10.1016/j.joep.2013.07.004>. URL <https://www.sciencedirect.com/science/article/pii/S0167487013000949>.
- [238] SciPy Community. *chi2\_contingency - SciPy v1.16.2 Manual*. SciPy Developers, 2023. URL [https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chi2\\_contingency.html#scipy.stats.chi2\\_contingency](https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.chi2_contingency.html#scipy.stats.chi2_contingency). Accessed: 2025-11-25.
- [239] SciPy Community. *association - SciPy v1.16.2 Manual*. SciPy Developers, 2023. URL <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.contingency.association.html>. Accessed: 2025-11-25.
- [240] Slimane Sefiane and Mohamed Benbouziane. Portfolio selection using genetic algorithm. *Journal of Applied Finance & Banking*, 2012.
- [241] Siddhartha Shakya, Alexander Brownlee, John McCall, François Fournier, and Gilbert Owusu. Deum—a fully multivariate eda based on markov networks. In *Exploitation of Linkage Learning in Evolutionary Algorithms*, pages 71–93. Springer, 2010.
- [242] Lloyd S Shapley et al. A value for n-person games. 1953.

- [243] Shubham Sharma, Jette Henderson, and Joydeep Ghosh. Certifai: A common framework to provide explanations and analyse the fairness and robustness of black-box models. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, pages 166–172. ACM, February 2020. doi: 10.1145/3375627.3375812. URL <http://dx.doi.org/10.1145/3375627.3375812>.
- [244] Farzaneh Shoeleh, Mahshid Majd, Ali Hamzeh, and Sattar Hashemi. Knowledge representation in learning classifier systems: A review. *arXiv preprint arXiv:1506.04002*, 2015.
- [245] Jeremy Silver. Yoochoose amazon book dataset. <https://github.com/jeremander/Yoochoose/tree/master/yoochoose>, 2026. GitHub repository (username: jeremander). Accessed: 2026-04-18.
- [246] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014. URL <https://arxiv.org/abs/1312.6034>.
- [247] Manjinder Singh, Alexander EI Brownlee, and David Cairns. Towards explainable metaheuristic: mining surrogate fitness models for importance of variables. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1785–1793, 2022.
- [248] Steven S Skiena. *The algorithm design manual*, volume 2. Springer, 2008.
- [249] Dylan Slack, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 180–186, 2020.
- [250] John Slaney and Toby Walsh. Backbones in optimization and approximation. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'01, pages 254–259, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558608125. doi: 10.5555/1642090.1642125.
- [251] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda ViÅgas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise, 2017. URL <https://arxiv.org/abs/1706.03825>.
- [252] Kacper Sokol and Julia E Vogt. What does evaluation of explainable artificial intelligence actually tell us? a case for compositional and contextual validation of xai building blocks. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–8, 2024.
- [253] Yuri N. Sotskov, Vjacheslav S. Tanaev, and Frank Werner and. Scheduling problems and mixed graph colorings. *Optimization*, 51(3):597–624, 2002. doi: 10.1080/0233193021000004994. URL <https://doi.org/10.1080/0233193021000004994>.
- [254] Charles Stan-Bishop, LC Barone, and Ronald Lyndon While. Visualisation of building blocks in evolutionary algorithms. In *2009 IEEE Congress on Evolutionary Computation*, pages 2431–2438. IEEE, 2009.
- [255] Ilia Stepin, Jose M Alonso, Alejandro Catala, and Martín Pereira-Fariña. A survey of contrastive and counterfactual explanation generation methods for explainable artificial intelligence. *Ieee Access*, 9:11974–12001, 2021.
- [256] Rolf Steyer and Werner Nagel. *Probability and conditional expectation: Fundamentals for the empirical sciences*. John Wiley & Sons, 2017.
- [257] Ashley Suh, Isabelle Hurley, Nora Smith, and Ho Chit Siu. Fewer than 1 URL <https://arxiv.org/abs/2503.16507>.
- [258] Emily Sullivan. Sides: Separating idealization from deceptive explanations in xai, 2024. URL <https://arxiv.org/abs/2404.16534>.
- [259] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks, 2017. URL <https://arxiv.org/abs/1703.01365>.

- [260] William R Swartout. Explaining and justifying expert consulting programs. In *Computer-assisted medical decision making*, pages 254–271. Springer, 1985.
- [261] El-Ghazali Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.
- [262] Robert F Tate. Correlation between a discrete and a continuous variable. point-biserial correlation. *The Annals of mathematical statistics*, 25(3):603–607, 1954.
- [263] Dirk Thierens. Linkage tree genetic algorithm: first results. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '10*, pages 1953–1958, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300735. doi: 10.1145/1830761.1830832. URL <https://doi.org/10.1145/1830761.1830832>.
- [264] Dirk Thierens. The linkage tree genetic algorithm. In *International Conference on Parallel Problem Solving from Nature*, pages 264–273. Springer, 2010.
- [265] Dirk Thierens and Peter Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th GECCO conference*, pages 617–624, New York, NY, USA, 2011. ACM.
- [266] Dirk Thierens and Peter Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th GECCO*, GECCO '11, pages 617–624, NY, USA, 2011. ACM. ISBN 9781450305570. doi: 10.1145/2001576.2001661.
- [267] Sarah L Thomson, Jason Adair, Alexander EI Brownlee, and Daan van den Berg. From fitness landscapes to explainable ai and back. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 1663–1667, 2023.
- [268] Chris Thornton. The building block fallacy. *Complexity International*, 4:1–1, 1997. ISSN 13200682.
- [269] Renato Tinós, Michal Przewozniczek, Darrell Whitley, and Francisco Chicano. Genetic algorithm with linkage learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 981–989, 2023.
- [270] Michael Trick. Graph coloring instances, 1994. URL <https://mat.tepper.cmu.edu/COLOR/instances.html>. Accessed: 2025-01-10.
- [271] Ryan J Urbanowicz and Will N Browne. *Introduction to learning classifier systems*. Springer, 2017.
- [272] Ryan J Urbanowicz and Jason H Moore. Learning classifier systems: a complete introduction, review, and roadmap. *Journal of Artificial Evolution and Applications*, 2009(1):736398, 2009.
- [273] Niki van Stein, Diederick Vermetten, Anna V. Kononova, and Thomas Bäck. Explainable benchmarking for iterative optimization heuristics, 2024. URL <https://arxiv.org/abs/2401.17842>.
- [274] Sahil Verma, Varich Boonsanong, Minh Hoang, Keegan Hines, John Dickerson, and Chirag Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. *ACM Computing Surveys*, 56(12):1–42, 2024.
- [275] Giulia Vilone and Luca Longo. Explainable artificial intelligence: a systematic review, 05 2020.
- [276] Giulia Vilone and Luca Longo. Notions of explainability and evaluation approaches for explainable artificial intelligence. *Information Fusion*, 76:89–106, 2021.
- [277] Marco Virgolin, Andrea De Lorenzo, Eric Medvet, and Francesca Randone. Learning a formula of interpretability to learn interpretable formulas. In Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann, editors, *Parallel Problem Solving from Nature – PPSN XVI*, pages 79–93, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58115-2.

- [278] Aidan Wallace, Alexander EI Brownlee, and David Cairns. Towards explaining metaheuristic solution quality by data mining surrogate fitness models for importance of variables. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 58–72. Springer, 2021.
- [279] Mathew J Walter, David J Walker, and Matthew J Craven. An explainable visualisation of the evolutionary search process. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1794–1802, 2022.
- [280] Sean P Walton, Ben J Evans, Alma AM Rahat, James Stovold, and Jakub Vincalek. Does mapping elites illuminate search spaces? a large-scale user study of map-elites applied to human-ai collaborative design. *arXiv preprint arXiv:2402.07911*, 2024.
- [281] Chen Wang, Yuhao Qian, and Seid Shaic. The applications of nature-inspired algorithms in logistic domains: a comprehensive and systematic review. *Arabian Journal for Science and Engineering*, 46(4):3443–3464, 2021.
- [282] Huanjing Wang, Qianxin Liang, John T Hancock, and Taghi M Khoshgoftaar. Feature selection strategies: a comparative analysis of shap-value and importance-based methods. *Journal of Big Data*, 11(1):44, 2024.
- [283] Yifei Wang. A comparative analysis of model agnostic techniques for explainable artificial intelligence. *Research Reports on Computer Science*, pages 25–33, 2024.
- [284] Rosina O Weber, Adam J Johs, Prateek Goel, and João Marques Silva. Xai is in trouble. *AI Magazine*, 45(3):300–316, 2024.
- [285] Dietrich Wettschereck. A hybrid nearest-neighbor and nearest-hyperrectangle algorithm. In *European Conference on Machine Learning*, pages 323–335. Springer, 1994.
- [286] David White. An overview of schema theory, 2014.
- [287] Dennis G Wilson, Sylvain Cussat-Blanc, Herv   Luga, and Julian F Miller. Evolving simple programs for playing atari games, 2018. URL <https://arxiv.org/abs/1806.05695>.
- [288] Stewart W Wilson. Classifiers that approximate functions. *Natural Computing*, 1(2):211–234, 2002.
- [289] Eyal Winter. The shapley value. *Handbook of game theory with economic applications*, 3:2025–2054, 2002.
- [290] In-On Wiratsin and Chaoyong Ragkhitwetsagul. Effectiveness of explainable artificial intelligence (xai) techniques for improving human trust in machine learning models: A systematic literature review. *IEEE Access*, 2025.
- [291] David H. Wolpert and William G. Macready. No free lunch theorems for search. In *No Free Lunch Theorems for Search*, 1995. URL <https://api.semanticscholar.org/CorpusID:12890367>.
- [292] Ka-Chun Wong. Evolutionary algorithms: Concepts, designs, and applications in bioinformatics. In *Handbook of Research on Advanced Hybrid Intelligent Techniques and Applications*, pages 190–215. IGI Global Scientific Publishing, 2016.
- [293] XGBoost Developers. Xgboost parameters, 2026. URL <https://xgboost.readthedocs.io/en/stable/parameter.html>.
- [294] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. Explainable ai: A brief survey on history, research areas, approaches and challenges. *Natural Language Processing and Chinese Computing*, 11839 LNAI:563–574, 2019. ISSN 16113349. doi: 10.1007/978-3-030-32236-6\_51.
- [295] Chen Ying-ping, Yu Tian-Li, S Kumara, and DE Goldberg. A survey of linkage learning techniques in genetic and evolutionary algorithms. *IlliGAL Report*, 2007.

- [296] Tian-Li Yu and David E. Goldberg. Conquering hierarchical difficulty by explicit chunking: Substructural chromosome compression. In *Proceedings of the 8th Annual GECCO Conference*, GECCO '06, pages 1385–1392, New York, NY, USA, 2006. ACM. ISBN 1595931864. doi: 10.1145/1143997.1144210.
- [297] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, 11(6): 712–731, 2007.
- [298] Yujia Zhang, Kuangyan Song, Yiming Sun, Sarah Tan, and Madeleine Udell. "why should you trust my explanation?" understanding uncertainty in lime explanations, 2019. URL <https://arxiv.org/abs/1904.12991>.
- [299] Weijie Zheng and Benjamin Doerr. Runtime analysis for the nsga-ii: Proving, quantifying, and explaining the inefficiency for many objectives. *IEEE Transactions on Evolutionary Computation*, 28(5):1442–1454, 2024. doi: 10.1109/TEVC.2023.3320278.
- [300] Ryan Zhou, Jaume Bacardit, Alexander Brownlee, Stefano Cagnoni, Martin Fyvie, Giovanni Iacca, John McCall, Niki van Stein, David Walker, and Ting Hu. Evolutionary computation and explainable ai: A roadmap to transparent intelligent systems. *arXiv preprint arXiv:2406.07811*, 2024.
- [301] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. *TIK report*, 103, 2001.



Part V

## APPENDICES

# A

## APPENDIX A: EXTENDED VERSION OF THE “MINING POTENTIALLY EXPLANATORY PATTERNS VIA PARTIAL SOLUTIONS

---

The following paper is the first publication resulting from this research, and contains various elements which have become outdated within the context of this thesis. Despite this, this work set out the core ideas of **PS**:

- using **\*-count**, **simplicity**, **mean-fitness** and **atomicity** as objectives,
- Using a static dataset, the  $P_{\text{Ref}}$ ,
- Notation used throughout this work

Here, **PS** mining is implemented as a local search in 3 different ways (akin to local search), in order to determine which strategy is the most effective. The results of this investigation then influenced the final implementation of the multi-objective search method discussed in Chapter 6. Another significant difference is the implementation of atomicity, which at this point was not bivariate yet, and instead it was based on the Mutual Information between a fixed variable and the rest of the pattern. This proved to be excessively slow and unstable, which is why it was replaced in our next publication.

Importantly, this work shows how the patterns expected for 3 benchmark problems were recovered using this initial **PS** mining methodology.

Additionally, this work shows a research direction which was not explored further but it is relevant to future work: the construction of new solutions by combining **PSs**. The algorithm which implements this, **pick-and-merge** is the basis of the **EDA** suggested in Section 11.3.4.

# Mining Potentially Explanatory Patterns via Partial Solutions

A proposed methodology and experimental results

GianCarlo A.P.I. Catalano

Alexander E.I. Brownlee

David Cairns

g.a.catalano@stir.ac.uk

alexander.brownlee@stir.ac.uk

dec@cs.stir.ac.uk

University of Stirling

Computing Science and Mathematics

Stirling, UK

John McCall

j.mccall@rgu.ac.uk

Robert Gordon University

National Subsea Centre

Aberdeen, UK

Russell Ainslie

russell.ainslie@bt.com

BT Technology

Applied Research

Ipswich, UK

## ABSTRACT

Genetic Algorithms have established their capability for solving many complex optimization problems. Even as good solutions are produced, the user's understanding of a problem is not necessarily improved, which can lead to a lack of confidence in the results. To mitigate this issue, explainability aims to give insight to the user by presenting them with the knowledge obtained by the algorithm.

In this paper we introduce Partial Solutions in order to improve the explainability of solutions to combinatorial optimization problems. Partial Solutions represent beneficial traits found by analyzing a population, and are presented to the user for explainability, but also provide an explicit model from which new solutions can be generated. We present an algorithm that assembles a collection of Partial Solutions chosen to strike a balance between high fitness, simplicity and atomicity. Experiments with standard benchmarks show that the proposed algorithm is able to find Partial Solutions which improve explainability at reasonable computational cost without affecting search performance.

## CCS CONCEPTS

• **Computing methodologies** → **Genetic algorithms**; • **Theory of computation** → *Models of computation*.

## KEYWORDS

Genetic Algorithms, Explainable AI (XAI), combinatorial optimization problems

## 1 INTRODUCTION

Genetic Algorithms (GA) are able to solve many kinds of optimization problems by producing high-quality results, which promotes their adoption even in critical applications [14, 30]. A problem presented by GAs, but also by Machine Learning, is that they act as black boxes which humans struggle to trust, despite the apparent quality of the outputs [3]. This mistrust is not unjustified: a GA's results can be sub-optimal for many reasons, especially due to stochastic behavior, incorrect fitness function implementations, unintended biases and design issues [30].

This is not a small problem, since user trust becomes increasingly important as GAs are introduced into high-stake applications [30]. The solution to this issue is **explainability**, where the user is presented with the knowledge synthesized by the algorithm to

solve the task. This can be used to debug the implementation, provide documentation for auditing purposes, manually adjusting and improve understanding of computer-generated solutions [30].

### 1.1 Optimization problems

The present paper will concentrate on combinatorial optimization problems, which can be particularly complex to solve despite their relatively simple construction. Many have been shown to be computationally hard, such as Graph Coloring [15], Knapsack [28] and resource allocation problems [22]. As such, these often have to be approached using metaheuristics such as GAs, but their internal mechanisms are often difficult for humans to follow, suggesting the need for techniques similar to XAI for Machine Learning [30].

There are many open questions which apply to optimization problems, that are also particularly suited for explainability:

- **Why is a solution good, or better than another?**
- **What characterizes good solutions?**
- **How can the global optima be detected?**

The approach we propose offers one way to address these questions. We introduce the concept of a *Partial Solution* (PS), an explicit decomposition of a positive trait identified in high-fitness solutions. A collection of PSs is assembled and acts as a model that is then sampled to generate new solutions. In doing so, the algorithm is inherently explainable, offering the ability for the end user to identify key components of the recommended solutions.

These Partial Solutions offer a novel way of providing both global and local explanations, but at the same time improve the search efficiency when solving the problem. We test whether it is possible to consistently find these Partial Solutions for some benchmark functions which have known underlying structures, in Section 5.2.1 and Section 5.2.2. Afterwards, the ability to solve problems efficiently is compared against a traditional GA and UMDA in Section 5.2.3.

## 2 RELATED WORK

We now summarize relevant research in the EC and optimization literature related to explainability.

### 2.1 Models within GAs and EDAs

Approaches which use a model to represent the problem, including surrogate fitness functions such as neural networks and decision trees, or the probabilistic models of Estimation of Distribution

Algorithms (EDA), have been a popular target for developing explainability functionality. This is because these models are designed to represent abstract information in order to accurately generalize the solution space. This knowledge can also be presented to the user for explainability purposes [1], drawing on the techniques developed for explainability in Machine Learning.

This paper makes use of related concepts from EDAs, as they present a fertile avenue for explainability applications that can be adapted to metaheuristics. Although traditional GAs lack explicit models, such structures can be added. For example, Brownlee [5] approximates the fitness function using a Markov Network, which can be then presented to the user, whereas Oh and Ahn [20] use a marginal product model for linkage learning, and Yu and Goldberg [31] use a hierarchical linkage model. Some methods can be argued to be using the population itself as a model, such as [10, 11], where the GA is analyzed in terms of the **trajectory** within the search space that the population takes during the evolution process. Similar to how an EDA generates its model from a population, the present work will analyze a population and produce a set of PSs, which then act as a model from which new solutions are generated. On the other hand, since this “model” is only built once and is not iterative, our approach is not a fully fledged EDA. Section 3.2 will give a formal definition of the proposed concepts, and Section 3.4 will review the novelty of this approach in more detail.

## 2.2 Backbones and building blocks

As discussed in [1], **backbones** are related to explainability because they give insight into patterns found in solutions. Backbones mostly relate to SAT problems, and consist of sub-configurations of values which are shared among the global optima. Mathematically, backbones and PSs share some similarities because they both represent commonalities within sets of solutions, but PSs are made with explainability in mind, and thus they are generated using different quality metrics and used very differently.

Both concepts are reminiscent of **schemata**, introduced by Holland in some early theoretical work to formalize the convergence properties of GAs [16, 29], but these theories were later shown to not match observed results [27]. It should be noted that these results are limited to Holland’s conjectures, and do not relate to the explicit construction of Partial Solutions or their use in explainability, which will be the central topics of this paper.

## 3 PARTIAL SOLUTIONS

### 3.1 Informal Introduction

To understand the behavior of the fitness function of a combinatorial optimization problem, one can look at patterns associated with higher fitness solutions. These may be higher-order interactions between groups of variables, e.g. ‘all variables in the group must be the same value’, but in the present paper the focus is on patterns where certain variables are fixed and others can vary (indicated using \*). For example  $0**1$  applies to  $0001$ ,  $0011$ ,  $0101$  and  $0111$ .

A collection of such patterns, found by analyzing a population, provides a succinct and expressive representation of what makes a solution “good” (as seen in Figure 1).

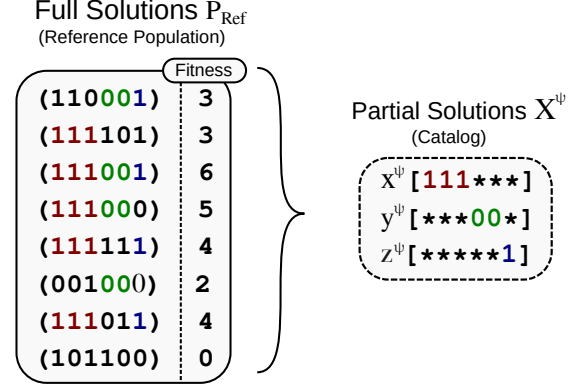


**Figure 1: The positive traits in a collection of full solutions  $P_{Ref}$  can be described by a collection of Partial Solutions  $X^\psi$**

In a real problem, PSs can offer useful insight into what variables are linked, and what values they should take. In an employee-allocation problem, for instance, these sub-configurations might indicate that it is beneficial to assign certain groups of employees to specific tasks. A solution such as 111111 might have its high fitness explained by the presence of 111\*\*\* and \*\*\*\*\*1, although 111001 is even better because of the presence of \*\*\*00\*.

Additionally, PSs might show the presence of positive traits that cannot coexist (e.g.,  $**00$  and  $*11*$ ), which can explain the presence of many near-global optima. The ability to find such disagreeing PSs can help in explaining situations where generally positive traits are not necessarily found in the global optima.

Partial solutions are an explicit decomposition of the problem, which has two benefits:

- Partial Solutions are **interpretable**:
  - **Global explanations**: describing the fitness landscape by finding simple and atomic Partial Solutions which are associated with high-fitness.
  - **Local explanations**: describing a solution by pointing out the Partial Solutions it contains.
- Partial Solutions **assist the search process**, because they act as a model that describes high-fitness regions of the solution space (similar to Probabilistic Graphical Models) from which solutions may be constructed (see Section 4.2).

### 3.2 Formal definition

Table 1 summarizes our notation. A Partial Solution  $x^\psi$  can be described as “a sub-configuration of parameter values which is associated with high-fitness solutions within a reference population”, where that reference population will be denoted as  $P_{Ref}$ .

In the same way that a full solution is a tuple of values  $x = (x_1, x_2, \dots, x_n)$ , a Partial Solution is  $x^\psi = (x_1^\psi, x_2^\psi, \dots, x_n^\psi)$  where each  $x_i^\psi$  can take any value that  $x_i$  can, as well as “\*”, the “any” value. A solution  $x$  is said to “contain” the Partial Solution  $x^\psi$  when they agree on the fixed values found in  $x^\psi$ .

$$\text{contains}(x, x^\psi) \Leftrightarrow (x_i^\psi = *) \vee (x_i^\psi = x_i) \forall i \in [1, n] \quad (1)$$

**Table 1: Notation**

| Symbol                                | Meaning                                                |
|---------------------------------------|--------------------------------------------------------|
| $F$                                   | Fitness function to be maximized                       |
| $F^\psi$                              | Fitness function for partial solutions                 |
| $n$                                   | The number of parameters in the solutions              |
| $a, b, \dots, x$                      | Variables used to denote a <i>full</i> solution        |
| $x_1, x_2, \dots, x_n$                | The parameters of $x$ , a full solution                |
| $a^\psi, b^\psi, \dots, x^\psi$       | Variable used to denote a <i>Partial</i> Solution      |
| $x_1^\psi, x_2^\psi, \dots, x_n^\psi$ | The parameters of $x^\psi$ , a <i>Partial</i> Solution |
| $P_{\text{Ref}}$                      | A collection of evaluated full solutions               |

Conversely, it is possible to find all the solutions in  $P_{\text{Ref}}$  which contain the Partial Solution  $x^\psi$ :

$$\text{observations}_{P_{\text{Ref}}}(x^\psi) = \{x \in P_{\text{Ref}} \mid \text{contains}(x, x^\psi)\} \quad (2)$$

For an optimization problem, within all of the possible PSs there is a small subset which is preferable for explainability, the **catalog**. These are the PSs which are optimal with respect to three metrics described in the next section: *simplicity*, *mean fitness* and *atomicity*.

### 3.3 Partial Solution metrics

It is preferable for a PS to have many of its parameters take the  $*$  value, since fewer fixed parameters improve interpretability. We define this metric to be:

$$\text{simplicity}(x^\psi) = |\{i \in [1, n] \mid x_i^\psi = *\}| \quad (3)$$

The simplest PS possible has no fixed variables, and consists entirely out of  $*$ 's. Other than being simple, a PS also needs to be associated with high-fitness solutions, which can be quantified by finding the mean fitness of its observations (cf. Equation (2)).

$$\text{meanFitness}(x^\psi) = \text{average}(F(\text{observations}_{P_{\text{Ref}}}(x^\psi))) \quad (4)$$

In order to have PSs assist with explainability, the previous two metrics are not sufficient. Assume  $x^\psi$  and  $y^\psi$  are optimal in different metrics, and let  $z^\psi$  be the union of their fixed values.  $z^\psi$  is likely to also be high-scoring, since it will have similar observations to its parents, and its simplicity might not be significantly worse than  $x^\psi$  and  $y^\psi$ , even though it is redundant compared to them.

This effect was described in Holland's Building Block Hypothesis as beneficial to the GA process, but for our purposes this is actually undesirable, since it gets in the way of interpretability due to all the redundancy present. To solve this problem, it will be necessary to define a third metric, which will encapsulate the idea that a PS should be "irreducible": **atomicity**.

For a Partial Solution  $x^\psi$ , its fixed-value elements  $x_j^\psi$  each contribute in some way to its average fitness. We define a normalized fitness function, remapped in the range  $[0, 1]$ :

$$F_{\text{Norm}}(x) = \frac{F(x) - m}{S} \quad (5)$$

(where  $m = \min_{x \in P_{\text{Ref}}} F(x)$ ,  $S = \sum_{x \in P_{\text{Ref}}} (F(x) - m)$ )

In order to measure the contribution of the variable  $x_k^\psi$  within  $x^\psi$ , define the following modifications of  $x^\psi$ :

- **isolate**: where only  $x_k^\psi$  is fixed
- **exclude**: where  $x_k^\psi$  is  $*$

And we define their "benefit" to be the sum of the normalized fitnesses of their observations.

$$\text{isolate}(x^\psi, k) = y^\psi \text{ s.t. } y_i^\psi = \begin{cases} x_i^\psi & \text{if } i = k \\ * & \text{otherwise} \end{cases} \quad (6)$$

$$\text{exclude}(x^\psi, k) = z^\psi \text{ s.t. } z_i^\psi = \begin{cases} x_i^\psi & \text{if } i \neq k \\ * & \text{otherwise} \end{cases} \quad (7)$$

$$\text{benefit}(a^\psi) = \sum_{x \in \text{observations}_{P_{\text{Ref}}}(a^\psi)} F_{\text{Norm}}(x) \quad (8)$$

Finally, we define the contribution of the variable  $x_k^\psi$  as:

$$\text{contribution}(x^\psi, k) = p_{AB} \cdot \log \left( \frac{p_{AB}}{p_{A*} \cdot p_{*B}} \right) \quad (9)$$

where  $p_{AB} = \text{benefit}(x^\psi)$ ,

$$p_{A*} = \text{benefit}(\text{isolate}(x^\psi, k)),$$

$$p_{*B} = \text{benefit}(\text{exclude}(x^\psi, k))$$

This is derived from mutual information, as used in DSMGA-II [17]. Continuing the example from Section 3.1, the Partial Solution 111\*\*1 will have its contributions calculated at  $k \in \{1, 2, 3, 6\}$ , and since independent groups are present the contribution scores will be low. For 111\*\*\*, the contributions are calculated at  $k \in \{1, 2, 3\}$ , which will all be greater than those in the previous example.

In the same way that a chain is as strong as its weakest link, the atomicity of  $x^\psi$  is defined as the *minimum* of its contributions, which ensures that all of the fixed parameters in the PS depend on each other and are essential.

$$\text{atomicity}(x^\psi) = \min \left\{ \text{contribution}(x^\psi, k) \mid k \in [1, n] \atop x_k^\psi \neq * \right\} \quad (10)$$

Each of these metrics is necessary in order to characterize the PSs which are best suited for explainability:

- **Simplicity** prioritizes more understandable PSs
- **Mean-fitness** promotes beneficial PSs
- **Atomicity** prevents redundant PSs

### 3.4 Novelty

Having defined Partial Solutions, we can now review related existing concepts to demonstrate where our novelty lies.

**Schemata** and **backbones** are similarly structured to Partial Solutions, but they are used very differently. **Schemata** are abstractions that were used for theoretical work on GAs, and are *implicitly sampled*, i.e. not meant to be explicitly instantiated. In contrast, our work does **instantiate** them, similar to other studies which use them to improve search efficiency [9, 13], whereas in this work they are designed mainly for explainability.

**Backbones** relate to SAT problems where multiple satisfactory solutions share a common sub-configuration. For most problems there is only one backbone, and it is preferred for it to have many

fixed variables in order to shrink the search space [24], whereas in this work the aims are reversed to improve interpretability.

Since in this work the PS catalog is treated as a model, EDAs are also an adjacent topic. In this sense, there are some related works: **Linkage learning** [2, 17] and **Family-of-Subsets** [25] aim to find which variables are interacting in the fitness function, and are then assembled into structures such as sets or cliques [21]. These structures only determine which variables are interacting, but their optimal value assignments are not specified, whereas PSs contain both. In other words, PSs are instances of the cliques / linked sets found by the aforementioned approaches.

## 4 OVERALL SYSTEM

The system that we propose consists of:

- **PS Miner**: obtains the PS catalog from a reference population, described in Section 4.1 and Algorithm 1
- **Pick & Merge**: an algorithm which forms solutions by combining elements from the catalog, described in Section 4.2 and Algorithm 2.

The following (one pass) sequence of operations combines them:

- (1) Generate a reference population  $P_{\text{Ref}}$ , and evaluate it
- (2) Apply the PS Miner on  $P_{\text{Ref}}$  to generate the PS catalog
- (3) Apply pick\_and\_merge on the PS catalog to obtain full solutions

The process of producing solutions offers the explainability benefits of PSs and maintains the ability to solve the original optimization problem (see Figure 2), all of which will be tested in Section 5.

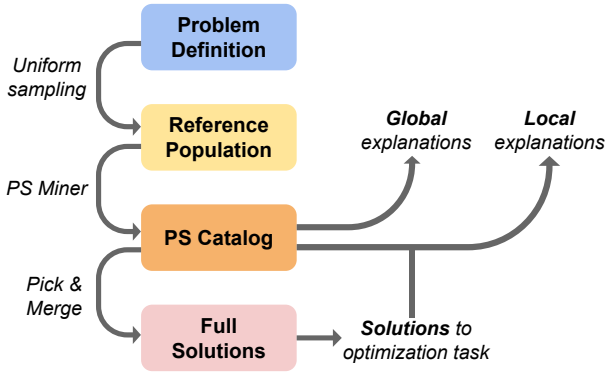


Figure 2: The proposed system, which uses the PS catalog for both explainability and to generate full solutions.

### 4.1 Finding Partial Solution Catalogs

The algorithm to find the PS catalog will be implemented using the metrics in Section 3.2, but the design of the search method involves some non-trivial design choices, which we will now describe.

**4.1.1 Reference Population.**  $P_{\text{Ref}}$  is used to evaluate the `meanFitness` and `atomicity` of PSs, and remains fixed throughout the PS Mining process. It's important for  $P_{\text{Ref}}$  to cover enough of the search space, and in the present work this was done using random uniform

sampling. Section 5.2.2 investigates the necessary size of  $P_{\text{Ref}}$  and whether it might be beneficial to evolve it.

Table 2: PS local search variants

| Variant                    | <code>get_local</code>                       | <code>get_init</code>                                                |
|----------------------------|----------------------------------------------|----------------------------------------------------------------------|
| <b>Simplification-only</b> | Simplifications of $x^\psi$                  | returns the top individuals in $P_{\text{Ref}}$ as Partial Solutions |
| <b>Specialization-only</b> | Specializations of $x^\psi$                  | returns the universal PS: <code>** ... **</code>                     |
| <b>Full-local</b>          | simplifications<br>$\cup$<br>specializations | The union of the above two methods                                   |

**4.1.2 Multiple objectives and  $F^\psi$ .** To simplify the system, **the three objectives are aggregated** into a single objective  $F^\psi$ . This is obtained by remapping each metric into the range  $[0, 1]$  and taking the mean, as seen in Algorithm 1.

Note that an evaluation of a full solution using  $F$  is a different operation from evaluating a Partial Solution with  $F^\psi$ , and the latter uses pre-calculated fitness values from  $P_{\text{Ref}}$  for efficiency.

**4.1.3 Exclusion archive.** A problem to be tackled when searching for the catalog is that the PS search space is large and contains many local optima clusters, which hinder the search process. Their presence is due to the fact that local optima for  $F$  also act as fully-set PS local optima for `meanFitness`, and their neighborhoods contain many similar individuals but with `*` values. In order to assist the algorithm, an **exclusion archive** based method is proposed, where the items selected in every iteration are put into an archive and are not allowed to reappear in the population again. In another sense, it marks which regions have been visited and prevents the population from revisiting them. The archive stores all of the best solutions seen throughout the iterations of the algorithm, and upon termination these are evaluated and the top scoring PSs are returned. This approach is implemented as shown in Algorithm 1, and it was tested as part of Section 5.2.1.

**4.1.4 Local search.** Knowing that the PS search space might contain many clustered optima, it is beneficial to include a form of local search. Given a PS  $x^\psi$ , its local neighborhood consists of

- **Simplifications**: one of the fixed variables is replaced by `*`
- **Specializations**: one of the `*` is replaced by a fixed value

Local search may use either or both of these, which determine the implementation of `get_local` and `get_init` in Algorithm 1. The three possible implementations will be tested as part of Section 5.2.1.

### 4.2 Pick & Merge algorithm

The PS catalog obtained via Algorithm 1 can be used as a model to construct high-quality full solutions. Algorithm 2 simply picks items from the PS catalog and merges them when possible. At the end, any remaining unfixed parameters are filled with random values, so that a full solution can be returned. Let  $X^\psi$  be the catalog found for a problem, and define the following operations, which are used in Algorithm 2:

**Algorithm 1:** Archive-based PS Miner

```

1 Def remap(values):
2   return  $\frac{\text{values} - \min(\text{values})}{\max(\text{values}) - \min(\text{values})}$ 
3 Def  $F^\psi(X^\psi)$ :
4    $M \leftarrow \text{remap}(\text{meanFitness}(X^\psi))$ 
5    $S \leftarrow \text{remap}(\text{simplicity}(X^\psi))$ 
6    $A \leftarrow \text{remap}(\text{atomicity}(X^\psi))$ 
7   return  $\{\text{avg}(m, s, a) \text{ for } m, s, a \in \text{zip}(M, S, A)\}$ 
8 Def top( $X^\psi$ , quantity):
9   sorted  $\leftarrow \text{sort}(X^\psi, \text{key} = F^\psi)$ 
10  return sorted[:quantity]

```

get\_init and get\_local are discussed in Section 4.1

```

12 Def mine_ps(get_init, get_local, pop_size, qty_ret):
13    $X^\psi \leftarrow \text{get\_init}()$ 
14   archive  $\leftarrow \{\}$ 
15   while termination_criteria_not_met do
16     selected  $\leftarrow \text{TournamentSelection}(X^\psi)$ 
17     localities  $\leftarrow \bigcup_{x^\psi \in \text{selected}} \text{get\_local}(x^\psi)$ 
18     archive  $\leftarrow \text{archive} \cup \text{selected}$ 
19      $X^\psi \leftarrow (X^\psi \cup \text{localities}) \setminus \text{archive}$ 
20      $X^\psi \leftarrow \text{top}(X^\psi, \text{pop\_size})$ 
21   return top(archive, qty_ret)

```

**Algorithm 2:** Pick-and-Merge algorithm

```

1 Def weighted_random_choice( $X^\psi$ ):
2   /* samples a random element of  $X^\psi$  using  $F^\psi$  as weights */
3 Def merge_from( $X^\psi$ , limit):
4   available  $\leftarrow \text{copy}(X^\psi)$ 
5    $x^\psi \leftarrow **..**$ 
6   added  $\leftarrow 0$ 
7   while available  $\neq \emptyset$  & added < limit do
8      $y^\psi \leftarrow \text{weighted\_random\_choice}(\text{available})$ 
9     available  $\leftarrow \text{available} \setminus \{y^\psi\}$ 
10    if mergeable( $x^\psi, y^\psi$ ) then
11       $x^\psi \leftarrow \text{merge}(x^\psi, y^\psi)$ 
12      added  $\leftarrow \text{added} + 1$ 
13  return  $x^\psi$ 
14 Def fill_gaps( $x^\psi$ ):
15   for  $x_i^\psi \in x^\psi$  do
16     if  $x_i^\psi = *$  then
17        $x_i^\psi \leftarrow \text{random.randrange}(\text{cardinalities}[i])$ 
18  return  $x^\psi$ 
19 Def generate_via_pick_and_merge( $X^\psi$ , merge_limit):
20  return fill_gaps(merge_from( $X^\psi$ , merge_limit))

```

$$\text{mergeable}(x^\psi, y^\psi) = \neg \exists i \text{ s.t. } (x_i^\psi \neq * \neq y_i^\psi \wedge x_i^\psi \neq y_i^\psi) \quad (11)$$

$$\text{merge}(x^\psi, y^\psi) = z^\psi \text{ s.t. } z_i^\psi = \begin{cases} x_i^\psi & \text{if } x_i^\psi \neq * \\ y_i^\psi & \text{if } y_i^\psi \neq * \\ * & \text{otherwise} \end{cases} \quad (12)$$

$$\text{to\_full}(x^\psi) = x \text{ s.t. } x_i = x_i^\psi \forall i \in [1, n] \quad (13)$$

(defined only when  $\neg \exists x_i^\psi = *$ )

The parameter `merge_limit` used in `pick_and_merge` determines how many PSs should be contained in a generated solution, which for this paper was set to  $\lceil \sqrt{n} \rceil$ . Note how this algorithm does not use the original fitness function, as it is assumed that  $X^\psi$  is representative of beneficial traits.

**5 EXPERIMENTS**

This section will present three research questions, **RQ1**, **RQ2** and **RQ3**, answered by the testing rounds **T1**, **T2** and **T3** respectively.

- **RQ1:** Which algorithm parameters are best suited for finding the PS catalog?
- **RQ2:** How is the ability to extract Partial Solutions from a reference population affected by its size and the amount of generations it has been evolved for?
- **RQ3:** Can Partial Solutions be used to obtain high-fitness full solutions? Our goal is increasing explainability, but the algorithm must still be able to solve the problem.

T1 and T2 will check which algorithms can produce the PS catalog effectively, which we assume will help explainability. For testing, we'll use benchmark problems with known catalogs, defined in Section 5.1. T3 will test the system's ability to solve the original optimization problem by checking how often it is able to find the global optima. The algorithms and the parameters being tested will be contained in Section 5.2, along with the description of the testing rounds. Finally, the results are presented and discussed in Section 6.

**5.1 Benchmark Problems**

Here we draw upon benchmarks previously used by the EDA community, as they have clearly defined structures that map well to graphical models, and present well-defined “target” PSs which we aim for our algorithms to find. The benchmark problems will need to be challenging enough to benefit from explainability, and hence “simpler” ones such as OneMax and BinVal will not be considered. Instead, problems which have dependencies between variables will be chosen, and to prevent any biases due to proximity in linked variables, the input variables will have their positions shuffled.

The **Royal Road** problem assigns a fitness to the input bit strings by dividing them into disjoint adjacent groups of  $k$  variables, and counting how many of them are entirely consisting of 1's [18]. The target PSs here are the individual groups, with their values all being set to 1. In this study  $k = 4$  (as can be seen in Figure 3), with 20 parameters in total, meaning that there are 5 target PSs.

A property of the Royal Road problem is that the target PSs are disjoint, which might be too trivial for the algorithm to solve. In order to increase the problem complexity, a variant called **Royal**

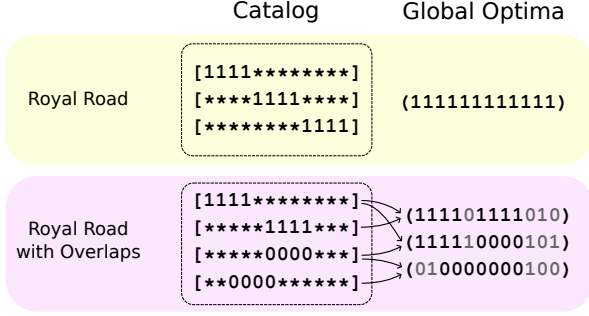


**Figure 3: Comparison of the RR and RRO problems, where RRO has more complex global optima**

**Road with Overlaps (RRO)** will be introduced, in which the groups are allowed to have overlaps and use 0's instead, as seen in Figure 3.

More formally, RRO is described by  $k$  (the size of each group),  $q$  (the number of groups) and  $l$  (the amount of bits in the total input). When  $l < q \cdot k$  the groups will be forced to overlap, which increases problem complexity. Let  $G^\psi$  be the targets, in the form of PSs, then the fitness function will be:

$$F_{RRO}(x) = |\{g^\psi \in G^\psi \text{ s.t. contains } (x, g^\psi)\}| \quad (14)$$

In this paper,  $k = 4$ ,  $q = 5$  and  $l = 15$  (guaranteeing overlap), with the groups being randomized for every run. A full solution containing all of the groups will often not be possible, and thus the task of finding PSs does not align with finding just the building blocks present in the global optima alone.

Another popular benchmark is **Trap-k**, in which each group of  $k$  bits is assigned a fitness which increases monotonically with the amount of 0's, but achieves its maximum when it contains only 1's, as seen in Equation (16) [12, 26].

$$F_{\text{Trap-k}}(x) = \sum_{i=0}^{(l/k)-1} F_{\text{Trap-k}}^{\text{sub}} \left( \sum_{j=k \cdot i}^{(k+1) \cdot i - 1} x_j \right) \quad (15)$$

$$F_{\text{Trap-k}}^{\text{sub}}(u) = \begin{cases} k & \text{if } u = k, \\ k - u - 1 & \text{if } u < k. \end{cases} \quad (16)$$

The problem was devised to be challenging for GAs, due to its deceptive fitness function, and it's been considered here to check whether deceptive PSs are a problem for our algorithms. Here,  $k = 5$  and  $l = 25$ , so that there will be 5 deceptive groups. The target PSs correspond to these groups, with all their parameters being set to 1.

## 5.2 Setup

When devising a PS mining algorithm, there are many potential sources of variability, including:

- The algorithm itself, due to selection operators
- The reference population, being randomly generated
- The problem definition, in the case of RRO

Due to this, whenever data is gathered about an algorithm configuration being executed on a certain problem, the run is executed 100 times, with a freshly generated reference population (and problem

**Table 3: PS Mining configurations**

| Algorithm    | Parameters                                                                                                                                       |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| Own method   | Population Size $\in \{50, 100, 150\}$                                                                                                           |
|              | Local Search $\in \left\{ \begin{array}{l} \text{Simplification-only,} \\ \text{Specialization-only,} \\ \text{Full-local} \end{array} \right\}$ |
| GA           | Uses Archive $\in \{\text{True}, \text{False}\}$                                                                                                 |
|              | One point mutation, rate of 0.075                                                                                                                |
| Hill Climber | Two point crossover, rate of 0.7                                                                                                                 |
|              | One point mutation, rate of 0.075                                                                                                                |
|              | Repeated trials, with each trial ending when a local optima is reached                                                                           |

**Table 4: Reference Population GA Parameters**

| Option    | Value                             |
|-----------|-----------------------------------|
| Selection | Tournament (size = 2), Elite = 2  |
| Mutation  | One point mutation, rate of 0.075 |
| Crossover | Two point crossover, rate of 0.7  |

definition for RRO). Additionally, the algorithms will be restricted to returning at most 50 PSs for every run (qty\_ret in Algorithm 1).

**5.2.1 T1.** Many configurations of algorithm parameters will be compared in their ability to **find PSs** efficiently. This will be done by measuring the number of  $F^\psi$  evaluations needed in order to find all the target PSs for the benchmark problems. The algorithms and the parameters that will be compared are shown in Table 3.

In each run,  $P_{\text{Ref}}$  will consist of  $10^4$  uniformly randomly generated full solutions. The algorithms will then be executed with the termination criterion being that they have found all of the target PSs, or that they have gone over  $10^5 F^\psi$  evaluations. Once the run terminates, it is considered **successful only if the output catalog contains all of the target PSs**.

**5.2.2 T2.** This round will contain a short analysis of how the ability to find the PS catalog is affected by the size of  $P_{\text{Ref}}$  and the number of generations it has been evolved for. Populations of sizes  $\in \{100, 200, 500, 1K, 2K, 5K, 10K\}$  are evolved for generations  $\in \{0, 10, 20, 50, 100, 200\}$ . The evolution process is done using a traditional GA, as described in Table 4 (derived from [7, 23]), and the **success criteria is the same as T1**.

**5.2.3 T3.** The ability for the system to **produce high-quality full solutions** will be tested by applying pick\_and\_merge as defined in Section 4.2. For evaluation budgets  $\in \{1K, 5K, 10K, 15K, 20K, 25K, 30K\}$ , three methods to generate full solutions will be compared: a traditional **GA** (as defined for T2), **UMDA** (both with population size of 150) and the method suggested in Section 4. The latter will use the settings selected from T1 to produce the catalog  $X^\psi$  via the **PS Miner** algorithm, by consuming a percentage ( $\in 10\%, 20\%, \dots, 90\%$ ) of the evaluation budget with  $F^\psi$  evaluations. The remaining evaluations are spent on  $F$  to evaluate solutions and construct  $P_{\text{Ref}}$ . The catalog obtained is passed to **pick\_and\_merge** to produce full

solutions. For each of these three methods, a run is considered **successful if it produces a global optima for the problem**.

## 6 RESULTS

### 6.1 T1

Table 5 shows the success rate for each PS Mining algorithm configuration, with the clear winner having the following properties:

- method: Archive-based
- population size: 150
- variant: Specialization-only

This configuration will be used in the tests that follow. In general, configurations with the archive enabled were more successful than their counterparts, with the specialization-only variants being the best across all problems. Additionally, the GA and Hill Climber were less successful, particularly with regard to the Trap-k problem.

**Table 5: Results for T1: success rates for different algorithm configurations (in bold are figures > 90%)**

| Search method       |           | Problem      |             |            |            |
|---------------------|-----------|--------------|-------------|------------|------------|
| Local search        | Pop. size | Uses archive | RR          | RRO        | Trap-k     |
| Simplification-only | 50        | True         | 0%          | 1%         | 0%         |
| Simplification-only | 100       |              | 0%          | 1%         | 0%         |
| Simplification-only | 150       |              | 0%          | 5%         | 0%         |
| Specialization-only | 50        |              | 64%         | 61%        | 47%        |
| Specialization-only | 100       |              | <b>96%</b>  | <b>93%</b> | 86%        |
| Specialization-only | 150       |              | <b>100%</b> | <b>94%</b> | <b>93%</b> |
| Full-local          | 50        | False        | 0%          | 36%        | 0%         |
| Full-local          | 100       |              | 0%          | 27%        | 0%         |
| Full-local          | 150       |              | 0%          | 31%        | 0%         |
| Simplification-only | 50        |              | 0%          | 0%         | 0%         |
| Simplification-only | 100       |              | 0%          | 0%         | 0%         |
| Simplification-only | 150       |              | 0%          | 0%         | 0%         |
| Specialization-only | 50        |              | 61%         | 12%        | 1%         |
| Specialization-only | 100       |              | <b>95%</b>  | 85%        | 10%        |
| Specialization-only | 150       |              | <b>100%</b> | <b>93%</b> | 30%        |
| Full-local          | 50        |              | 0%          | 0%         | 0%         |
| Full-local          | 100       |              | 0%          | 6%         | 0%         |
| Full-local          | 150       |              | 0%          | 18%        | 0%         |
| GA                  | 50        | -            | 0%          | 0%         | 0%         |
| GA                  | 100       | -            | 0%          | 0%         | 0%         |
| GA                  | 150       | -            | 0%          | 0%         | 0%         |
| Hill-climber        | -         | -            | 28%         | 31%        | 0%         |

### 6.2 T2

The data in Table 6 suggests that the best  $P_{Ref}$  to find the catalog is unevolved, with its minimum size =  $10^4$  for the tested problems. A possible explanation for this effect is the phenomenon of “*hitch-hiking*”, where sub-optimal traits spread throughout a population due to their co-occurrence with optimal traits. This effect was hypothesized by Mitchell et al. [19] upon observing similar results for Royal functions, where methods which rely on schemata (such as [4]) perform worse when applied on pre-evolved populations.

### 6.3 T3

Table 7 and Figure 4 show three important facts:

- (1) The PS catalog mediated approach is able to consistently find the global optima for all of the problems, when between 20% and 60% being the recommended proportion of the budget to dedicate to  $F^\psi$  evaluations.

- (2) for the RR and RRO problems, the GA and the PS catalog mediated approach perform similarly when around 50% of the evaluations are used for  $F^\psi$ .
- (3) For Trap-k, only the proposed method was able to find the global optima consistently, whereas the GA and UMDA almost always got stuck in the local optima.

These results give evidence that the proposed methodology can solve optimization problems with similar performance to a GA, improving on Trap-k, while adding explainability.

## 7 CONCLUSION

The algorithms presented in this paper describe a novel technique for explainability, by using Partial Solutions. The PS catalog provides a **global explanation** of the fitness landscape by presenting beneficial sub-configurations, while **local explanations** are obtained by checking which PSs are found in a solution. Moreover, the catalog acts similar to a model in an EDA, from which new solutions can be sampled and thus solving the original optimization problem. The testing rounds showed the following results:

- **RQ1:** The algorithm which is most apt for finding the PS catalog in the benchmarks was the Specialization-only local search, using the exclusion archive.
- **RQ2:** To obtain the PS catalog, it is better for  $P_{Ref}$  to be a large unevolved population
- **RQ3:** The system is consistently able to get the global optima for the benchmark problems by first generating a PS catalog and then applying `pick_and_merge`. This suggests that the PS catalog can capture important information about the fitness landscape. The recommended proportion of  $F^\psi$  evaluations is between 20% and 60%.

It should be noted that the PS Miner algorithms implemented in this work are not as efficient as they could be, since  $F^\psi$  involves complex calculations, and experimentation indicates this to be between 50 and 100 times slower than  $F$  for  $|P_{Ref}| = 10^4$ . Therefore the explainability of Partial Solutions still comes at a computational cost, although this might be partially countered by an improved chance at finding global optima (as seen for Trap-k). The source code for the system can be viewed at [8].

## 8 FUTURE WORK

The PS catalog represents beneficial traits for the fitness function, and thus they can act as a **model** that could be used to implement a surrogate fitness function (similar to [6]) or improved GA operators (similar to Optimal Mixing in [26]). Overall, the system could be readapted to be a full EDA, for example by feeding the outputs of `pick_and_merge` into  $P_{Ref}$ .

The problems discussed in this document are admittedly simpler than **real-world problems**, and therefore it would be useful to see how well the PSs perform with some **inherently inelegant problems**, such as rostering and scheduling tasks. In particular, there will be cases where interpretable PSs are not always present or difficult to find. This will be investigated in problems without clean-cut cliques, where the explainability of PSs will be explored more thoroughly.

As mentioned in Section 3.2, a PS may be **generalized to higher-order** relationships. This will both greatly improve the descriptive

Table 6: PS catalog success rate for T2, in bold are figures &gt; 90% (success is finding all of the target Partial Solutions)

| RR             |             |             |             |             |             |             | RRO            |            |     |     |            |            |     | Trap-k         |            |     |     |     |     |     |
|----------------|-------------|-------------|-------------|-------------|-------------|-------------|----------------|------------|-----|-----|------------|------------|-----|----------------|------------|-----|-----|-----|-----|-----|
| Gen's          | 0           | 10          | 20          | 50          | 100         | 200         | Gen's          | 0          | 10  | 20  | 50         | 100        | 200 | Gen's          | 0          | 10  | 20  | 50  | 100 | 200 |
| $P_{Ref}$ Size |             |             |             |             |             |             | $P_{Ref}$ Size |            |     |     |            |            |     | $P_{Ref}$ Size |            |     |     |     |     |     |
| 100            | 0%          | 0%          | 0%          | 0%          | 0%          | 0%          | 100            | 4%         | 0%  | 0%  | 0%         | 0%         | 0%  | 100            | 0%         | 0%  | 0%  | 0%  | 0%  | 0%  |
| 200            | 0%          | 0%          | 0%          | 0%          | 0%          | 0%          | 200            | 22%        | 8%  | 9%  | 4%         | 5%         | 4%  | 200            | 0%         | 0%  | 0%  | 0%  | 0%  | 0%  |
| 500            | 52%         | 23%         | 22%         | 12%         | 23%         | 23%         | 500            | 66%        | 55% | 47% | 50%        | 45%        | 46% | 500            | 0%         | 0%  | 0%  | 0%  | 0%  | 0%  |
| 1000           | <b>90%</b>  | 77%         | 80%         | 75%         | 80%         | 82%         | 1000           | 83%        | 82% | 77% | 75%        | 81%        | 78% | 1000           | 0%         | 0%  | 0%  | 0%  | 0%  | 0%  |
| 2000           | <b>100%</b> | <b>99%</b>  | <b>98%</b>  | <b>99%</b>  | <b>98%</b>  | <b>99%</b>  | 2000           | 89%        | 88% | 89% | 87%        | 87%        | 84% | 2000           | 2%         | 0%  | 0%  | 0%  | 0%  | 0%  |
| 5000           | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | 5000           | 89%        | 86% | 88% | <b>91%</b> | <b>90%</b> | 89% | 5000           | 38%        | 11% | 4%  | 3%  | 3%  | 6%  |
| 10000          | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | 10000          | <b>94%</b> | 89% | 89% | 89%        | 89%        | 89% | 10000          | <b>93%</b> | 51% | 60% | 53% | 49% | 60% |

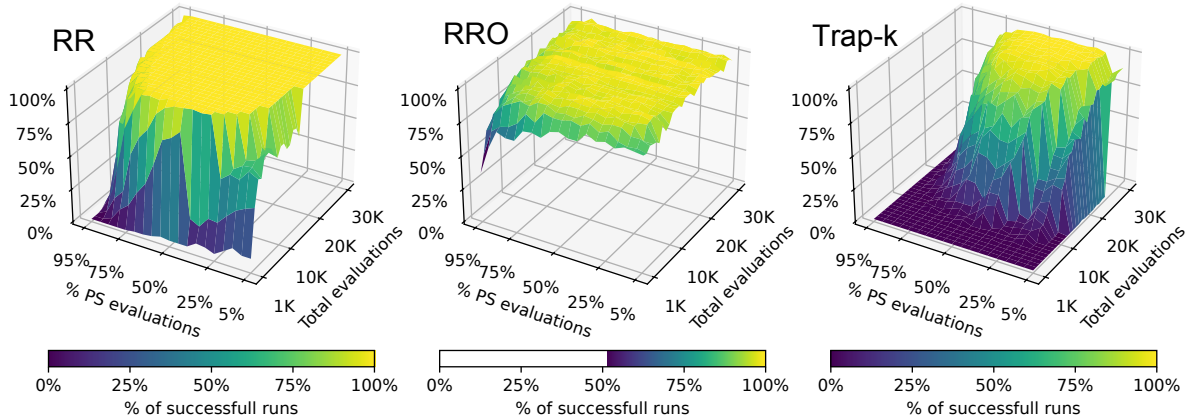


Figure 4: (Relating to T3) Mean fitness obtained by dedicating some of the evaluation budget to PS catalog mining.

Table 7: Global optima success rate for T3, in bold are figures &gt; 90% (success is finding a global optima)

| RR     |     |             |             |             |             |             |             | RRO    |            |             |            |             |            |            |            | Trap-k |    |    |     |     |            |            |             |
|--------|-----|-------------|-------------|-------------|-------------|-------------|-------------|--------|------------|-------------|------------|-------------|------------|------------|------------|--------|----|----|-----|-----|------------|------------|-------------|
| Eval's | 1K  | 5K          | 10K         | 15K         | 20K         | 25K         | 30K         | Eval's | 1K         | 5K          | 10K        | 15K         | 20K        | 25K        | 30K        | Eval's | 1K | 5K | 10K | 15K | 20K        | 25K        | 30K         |
| P&M    |     |             |             |             |             |             |             | P&M    |            |             |            |             |            |            |            | P&M    |    |    |     |     |            |            |             |
| 10%    | 9%  | 71%         | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | 10%    | 86%        | <b>96%</b>  | <b>96%</b> | <b>97%</b>  | <b>98%</b> | <b>97%</b> | <b>98%</b> | 10%    | 0% | 0% | 0%  | 53% | 71%        | 82%        | 81%         |
| 20%    | 8%  | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | 20%    | 84%        | <b>95%</b>  | <b>97%</b> | <b>99%</b>  | <b>99%</b> | <b>98%</b> | <b>97%</b> | 20%    | 0% | 0% | 13% | 39% | <b>97%</b> | <b>99%</b> | <b>100%</b> |
| 30%    | 6%  | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | 30%    | 85%        | <b>97%</b>  | <b>97%</b> | <b>98%</b>  | <b>98%</b> | <b>98%</b> | <b>96%</b> | 30%    | 0% | 0% | 5%  | 79% | <b>99%</b> | <b>99%</b> | <b>99%</b>  |
| 40%    | 3%  | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | 40%    | 81%        | <b>97%</b>  | <b>97%</b> | <b>99%</b>  | <b>96%</b> | <b>98%</b> | <b>96%</b> | 40%    | 0% | 0% | 21% | 87% | <b>95%</b> | <b>99%</b> | <b>100%</b> |
| 50%    | 3%  | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | 50%    | 84%        | <b>98%</b>  | <b>98%</b> | <b>98%</b>  | <b>96%</b> | <b>97%</b> | <b>97%</b> | 50%    | 0% | 0% | 14% | 74% | <b>92%</b> | <b>99%</b> | <b>100%</b> |
| 60%    | 2%  | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | 60%    | 76%        | <b>96%</b>  | <b>98%</b> | <b>99%</b>  | <b>98%</b> | <b>98%</b> | <b>97%</b> | 60%    | 0% | 0% | 6%  | 52% | 89%        | <b>96%</b> | <b>99%</b>  |
| 70%    | 1%  | <b>98%</b>  | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | 70%    | 77%        | <b>94%</b>  | <b>92%</b> | <b>97%</b>  | <b>96%</b> | <b>97%</b> | <b>96%</b> | 70%    | 0% | 0% | 4%  | 38% | 72%        | 89%        | <b>93%</b>  |
| 80%    | 0%  | 88%         | <b>98%</b>  | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | 80%    | 68%        | <b>92%</b>  | <b>93%</b> | <b>99%</b>  | <b>95%</b> | <b>96%</b> | <b>96%</b> | 80%    | 0% | 0% | 0%  | 11% | 24%        | 58%        | 71%         |
| 90%    | 0%  | 17%         | 79%         | <b>95%</b>  | <b>100%</b> | <b>100%</b> | <b>100%</b> | 90%    | 56%        | 81%         | 84%        | <b>96%</b>  | <b>96%</b> | <b>96%</b> | <b>96%</b> | 90%    | 0% | 0% | 0%  | 0%  | 4%         | 3%         | 8%          |
| GA     | 13% | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | <b>100%</b> | GA     | <b>98%</b> | <b>100%</b> | <b>99%</b> | <b>100%</b> | <b>99%</b> | <b>98%</b> | <b>98%</b> | GA     | 0% | 0% | 0%  | 0%  | 0%         | 0%         | 0%          |
| UMDA   | 0%  | 3%          | 1%          | 0%          | 0%          | 0%          | 0%          | UMDA   | 73%        | 77%         | 69%        | 68%         | 63%        | 73%        | 71%        | UMDA   | 0% | 0% | 0%  | 0%  | 0%         | 0%         | 0%          |

power of PSs, but also greatly increase the complexity of the search methods.  $F^\psi$  could also be extended by adding more metrics, such as novelty, robustness (to changes in the parameter values with respect to fitness), or constraint satisfaction.

## ACKNOWLEDGMENTS

Work funded under a PhD project supported by The Data Lab and BT Group plc. Many thanks go to the patience of my supervisors.

## REFERENCES

- [1] Jaume Bacardit, Alexander E. I. Brownlee, Stefano Cagnoni, Giovanni Iacca, John McCall, and David Walker. 2022. The Intersection of Evolutionary Computation and Explainable AI. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Boston, Massachusetts) (GECCO '22). Association for Computing Machinery, New York, NY, USA, 1757–1762. <https://doi.org/10.1145/3520304.3533974>
- [2] Shumeet Baluja and Scott Davies. 1997. Using Optimal Dependency-Trees for Combinational Optimization. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML '97)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 30–38.
- [3] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bernetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82–115. <https://doi.org/10.1016/j.inffus.2019.12.012>
- [4] D.F. Brown, A.B. Garmendia-Doval, and J.A.W. McCall. 2002. Markov Random Field Modelling of Royal Road Genetic Algorithms. In *Artificial Evolution*, Pierre

- Collet, Cyril Fonlupt, Jin-Kao Hao, Evelyne Lutton, and Marc Schoenauer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 65–76.
- [5] Alexander E.I. Brownlee. 2016. Mining Markov Network Surrogates for Value-Added Optimisation. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion* (Denver, Colorado, USA) (GECCO '16 Companion). Association for Computing Machinery, New York, NY, USA, 1267–1274. <https://doi.org/10.1145/2908961.2931711>
  - [6] Alexander Edward Ian Brownlee. 2009. *Multivariate Markov networks for fitness modelling in an estimation of distribution algorithm*. Ph. D. Dissertation. "Robert Gordon University".
  - [7] Alexander E. I. Brownlee, Olivier Regnier-Coudert, John A. W. McCall, and Stewart Massie. 2010. Using a Markov network as a surrogate fitness function in a genetic algorithm. In *IEEE Congress on Evolutionary Computation*. IEEE, New York City, USA, 1–8. <https://doi.org/10.1109/CEC.2010.5586548>
  - [8] Giancarlo Catalano. 2024. *PS Assisted Explainability*. [https://github.com/Giancarlo-Catalano/PS\\_Minimal\\_Showcase](https://github.com/Giancarlo-Catalano/PS_Minimal_Showcase)
  - [9] Jack McKay Fletcher and Thomas Wennekers. 2017. A natural approach to studying schema processing. arXiv:1705.04536 [cs.NE]
  - [10] Martin Fyvie, John A.W. McCall, and Lee A. Christie. 2021. Non-deterministic solvers and explainable AI through trajectory mining. *SICSA XAI Workshop 2021* 2894 (6 2021), 75–78. <https://rgu-repository.worktribe.com/output/1395881https://rgu-repository.worktribe.com/output/1395881.abstract>
  - [11] Martin Fyvie, John A. W. McCall, Lee A. Christie, Alexander E. I. Brownlee, and Manjinder Singh. 2021. Towards explainable meta-heuristics: Feature extraction from trajectory mining. *Expert Systems* n/a, n/a (2021), e13494. <https://doi.org/10.1111/exsy.13494> arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1111/exsy.13494>
  - [12] David E. Goldberg. 1989. Genetic algorithms and Walsh functions: Part 2, Deception and its analysis. *Complex systems* 3 (1989), 153–171.
  - [13] David E. Goldberg, Kumara Sastry, and Yukio Ohsawa. 2003. *Discovering Deep Building Blocks for Competent Genetic Algorithms Using Chance Discovery via KeyGraphs*. Springer, Berlin, Heidelberg, Tokyo, Japan, 276–301. [https://doi.org/10.1007/978-3-662-06230-2\\_19](https://doi.org/10.1007/978-3-662-06230-2_19)
  - [14] Katja Grace, John Salvatier, Allan Dafoe, Baobao Zhang, and Owain Evans. 2018. Viewpoint: When Will AI Exceed Human Performance? Evidence from AI Experts. *Journal of Artificial Intelligence Research* 62 (7 2018), 729–754. <https://doi.org/10.1613/JAIR.1.11222>
  - [15] Juris Hartmanis. 2006. Computers and Intractability: A Guide to the Theory of NP-Completeness (Michael R. Garey and David S. Johnson). [https://doi.org/10.1137/1024022\\_24](https://doi.org/10.1137/1024022_24) (7 2006), 90–91. Issue 1. <https://doi.org/10.1137/1024022>
  - [16] John H. Holland. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, Massachusetts, USA. <https://doi.org/10.7551/mitpress/1090.001.0001>
  - [17] Shih-Huan Hsu and Tian-Li Yu. 2015. Optimization by pairwise linkage detection, incremental linkage set, and restricted/back mixing: DSMGA-II. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. Association for Computing Machinery, New York, NY, USA, 519–526.
  - [18] Melanie Mitchell, Stephanie Forrest, and John Holland. 1992. The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance. *European Conference on Artificial Life* 1 (11 1992), 23–33.
  - [19] Melanie Mitchell, John H. Holland, and Stephanie Forrest. 1993. When will a genetic algorithm outperform hill climbing?. In *Proceedings of the 6th International Conference on Neural Information Processing Systems* (Denver, Colorado) (NIPS'93). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 51–58.
  - [20] Sanghoun Oh and Chang Wook Ahn. 2023. Evolutionary Approach for Interpretable Feature Selection Algorithm in Manufacturing Industry. *IEEE Access* 11 (2023), 46604–46614. <https://doi.org/10.1109/ACCESS.2023.3274490>
  - [21] Martin Pelikan, David E Goldberg, and Fernando G Lobo. 2002. A survey of optimization by building and using probabilistic models. *Computational optimization and applications* 21 (2002), 5–20.
  - [22] L. Rabiner. 1984. Combinatorial optimization: Algorithms and complexity. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32 (12 1984), 1258–1259. Issue 6. <https://doi.org/10.1109/TASSP.1984.1164450>
  - [23] Siddhartha Shakya, John McCall, and Deryck Brown. 2005. Using a Markov Network Model in a Univariate EDA: An Empirical Cost-Benefit Analysis. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation* (Washington DC, USA) (GECCO '05). Association for Computing Machinery, New York, NY, USA, 727–734. <https://doi.org/10.1145/1068009.1068130>
  - [24] John Slaney and Toby Walsh. 2001. Backbones in Optimization and Approximation. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1* (Seattle, WA, USA) (IJCAI'01). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 254–259. <https://doi.org/10.5555/1642090.1642125>
  - [25] Dirk Thierens and Peter AN Bosman. 2011. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. Association for Computing Machinery, New York, NY, USA, 617–624.
  - [26] Dirk Thierens and Peter A.N. Bosman. 2011. Optimal Mixing Evolutionary Algorithms. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation* (Dublin, Ireland) (GECCO '11). Association for Computing Machinery, New York, NY, USA, 617–624. <https://doi.org/10.1145/2001576.2001661>
  - [27] Chris Thornton. 1997. The building block fallacy. *Complexity International* 4 (1997), 1–1.
  - [28] Pamela H. Vance. 2006. Knapsack Problems: Algorithms and Computer Implementations (S. Martello and P. Toth). [https://doi.org/10.1137/1035174\\_35](https://doi.org/10.1137/1035174_35) (7 2006), 684–685. Issue 4. <https://doi.org/10.1137/1035174>
  - [29] David White. 2014. An Overview of Schema Theory. arXiv:1401.2651 [cs.NE]
  - [30] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. 2019. Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 11839 LNAI (2019), 563–574. [https://doi.org/10.1007/978-3-030-32236-6\\_51/FIGURES/12](https://doi.org/10.1007/978-3-030-32236-6_51/FIGURES/12)
  - [31] Tian-Li Yu and David E. Goldberg. 2006. Conquering Hierarchical Difficulty by Explicit Chunking: Substructural Chromosome Compression. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (Seattle, Washington, USA) (GECCO '06). Association for Computing Machinery, New York, NY, USA, 1385–1392. <https://doi.org/10.1145/1143997.1144210>

# B

## APPENDIX B: ALGORITHMS FOR VARIOUS METRICS

---

### B.1 ON THE REPRESENTATION OF PARTIAL SOLUTIONS

In my work, **PSs** are implemented as a **numpy** array where the **\*** wildcard is represented by the value  $-1$ . All other values are integers starting from zero. Although there are other formats to represent patterns, this occupies the least space in memory, which is a concern when there is an entire population of them, and each is potentially hundreds of variables long. Internally, these are stored as **int8**, because we never expect to find a variable with cardinality over 255.

### B.2 FINDING SOLUTIONS WHICH MATCH A PATTERN, AND THEIR FITNESS

Finding the solutions which match a pattern is the core idea of this entire research, and it is done for every pattern considered in the **PS** mining search process.

The solutions and their fitnesses are stored in a data structure called the  $P_{\text{Ref}}$ . This is just an array of solutions and their precomputed fitness values. This is roughly implemented as follows:

```
class PRef:
    solutions_array: np.ndarray
    fitness_array: np.ndarray
    search_space_cardinalities: np.ndarray

    def get_indices_of_matching_solutions(ps: PS):
        # trivial and slow implementation

    def get_fitnesses_of_matching_solutions(ps: PS):
        indices = self.get_indices_of_matching_solutions(ps)
        return self.fitness_array[indices]
```

Instead of finding the matching solutions directly, returning their indices allows for much more flexibility (since sometimes the solutions are needed, and other times the fitnesses are).

The method `get_indices_of_matching_solutions` is the bottleneck of the search process, and therefore it needs to be optimized. A trivial version would be as follows:

---

**Algorithm 21** Trivial Matching Algorithm

---

```
1: function TRIVIAL_GET_INDICES( $x^\psi$ )
2:   non_wildcards  $\leftarrow x^\psi \neq -1$  ▷ Boolean array
3:   matches  $\leftarrow []$ 
4:   for  $index, solution \in self.solutions\_array$  do
5:     differences  $\leftarrow (solution \neq x^\psi)$ 
6:     mismatches  $\leftarrow np.logical\_and(non\_wildcards, differences)$ 
7:     if not  $np.any(mismatches)$  then
8:       matches.append( $index$ )
9:     end if
10:  end for
11:  return matches
12: end function
```

---

Although **numpy** hides some of the logic, checking every single solution to see if it matches the pattern is rather slow.

There is a simple optimisation: if for example it is known that the pattern starts with a 1, then we can exclude any solutions which do not start with a one and reduce the search over that set. This can be generalized into the following process.

PRE-COMPUTE INDICES First, we construct a dictionary that maps every pair of  $(variable, value)$  to an unordered set of indices.

`precomputed_indices[(a, b)]` is the indices of solutions where `solution[a] = b`.

---

#### Algorithm 22 Inverted Index Preparation

---

```

1: precomputed_indices ← defaultdict(set)
2: for index, solution ∈ self.solutions_array do
3:     for variable, value ∈ solution do
4:         precomputed_indices[(variable, value)].add(index)
5:     end for
6: end for

```

---

We can now utilize set intersections to find matches efficiently. The algorithm below uses those precomputed indices to gradually narrow down the set of possibly matching solutions.

---

#### Algorithm 23 Optimized Set-Intersection Matching

---

```

1: function GET_MATCHING_INDICES( $x^\psi$ )
2:     ▷ precomputed_indices is pre-computed once
3:     if  $x^\psi$ .is_empty() then
4:         return set(range(len(self.solutions_array)))
5:     else
6:         sets_to_intersect ← []
7:         for var, val ∈ enumerate( $x^\psi$ ) do
8:             if val ≠ -1 then
9:                 possible_indices ← precomputed_indices[(var, val)]
10:                sets_to_intersect.append(possible_indices)
11:            end if
12:        end for
13:        return set.intersection(*sets_to_intersect)
14:    end if
15: end function

```

---

This implementation relies on the fact that in Python, the `set` class is implemented in C and is therefore much more efficient than most of the code that could have been written manually. The internal implementation of `set.intersection` will start from the smallest set and exclude as many items as soon as possible from the intersection.

Moreover, the solutions in  $P_{\text{notmatch}}$  can be efficiently calculated as the set difference between all the possible indices and the matching indices.

### B.3 ALGORITHM TO FIND SOLUTIONS WHICH DIFFER BY A SINGLE VARIABLE

For Dominance-based linkage, it is necessary to have a collection of pairs of solutions, such that these solutions differ by a *single variable*. From now on, the term “Hamming-distance-1” will be used for clarity.

The algorithm described here will find any such pairs in a given list of solutions in an efficient manner.

#### B.3.1 Naïve Approach

Firstly, let us consider the naïve approach: check every single pair of solutions.

```

def find_hamming_distance_1_pairs_naive(solutions_array):
    # pairs_by_difference[v] represents all the pairs that
    # differ by variable v
    pairs_by_difference = [[] for _ in range(L)]

```

```

for index_a, solution_a in enumerate(solutions_array):
    for index_b in range(index_a+1, len(solutions_array)):
        solution_b = solutions_array[index_b]

        quantity_of_differences = np.sum(solution_a != solution_b)
        if quantity_of_differences == 1:
            # find the position of the difference
            position_of_difference = None
            for i in range(L):
                if (solution_a[i] != solution_b[i]):
                    position_of_difference = i
                    break
            # then add the pair to the correct list
            pairs_by_difference[position_of_difference].append(
                (index_a, index_b)
            )

```

If  $N$  is the quantity of solutions in our dataset, this algorithm will make  $N \times (N - 1)/2$  pairwise checks, and each check is based on  $L$  comparisons. Therefore, the complexity will roughly be  $L \times N^2$  (because there are  $L \times N \times (N - 1)/2$  operations).

Moreover, take notice of the format of the output: this is the format that is most suitable for later analysis. For example, if `pairs_by_difference` is `{(4, 5), (8, 10), ...}`, it means that the solutions at indices 4 and 5 in the `solutions_array` differ only by the first variable, and the same applies to 8 and 10.

### B.3.2 Optimized Approach using Hash Collisions

The approach used here is based on *hash collisions*. Imagine a hash function which considers all the variables in the solutions except for the last one. If two solutions have the same hash value, then they are very likely to be different only in that last value. If a different hash function which ignores one variable is implemented for every variable, then we have a fast approach of quickly excluding solutions from our pairwise checking.

The algorithm proceeds as follows:

1. Precompute helper values, which are then used to calculate different hash values efficiently.
2. Let `collected_solutions` be a list of dictionaries (specifically `defaultdict(set)`).
  - `collected_solutions[a][hash_value]` is the set of indices of solutions whose hash value (ignoring variable  $a$ ) equals `hash_value`.
3. For every solution, and its index:
  - For every variable  $v$  from 0 to  $L - 1$ :
    - `hash_value` = hash value that ignores  $v$  for solution.
    - `collected_solutions[v][hash_value].append(index)`.
4. Initialise `list_of_lists_of_pairs` to be `[list() for _ in range(L)]`.
  - `list_of_lists_of_pairs[v]` is a list of pairs. Each pair is the indices of two solutions that differ only by  $v$ .
5. For every variable  $v$  from 0 to  $L - 1$ :
  - For every `hash_collision_group` in `collected_solutions.values()`:
    - A `hash_collision_group` is a set of indices of solutions that are likely to differ by only one value. A pair needs to differ by exactly one variable (not 0, not more than 1). The suitable pairs are added to `list_of_lists_of_pairs[v]`.

The hash values are calculated using a base, powers, and the modulo operator, in such a way that the hash values can be derived from each other. The base and modulo are chosen to be big prime numbers to avoid the risk of unwanted collisions, but their specific value is not too important.

The final algorithm is shown below.

```

def get_paired_samples_from_observations(solutions_array,
                                         base=911382323,
                                         mod=972663749):

    N, L = solutions_array.shape

    # Precompute powers of base
    p_pow = np.ones(L + 1, dtype=np.int64)
    for i in range(1, L + 1):
        p_pow[i] = (p_pow[i - 1] * base) % mod

    # Prefix hashes: prefix_hash[i, k] = hash of solutions_array[i, :k]
    prefix_hash = np.zeros((N, L + 1), dtype=np.int64)
    for k in range(1, L + 1):
        prefix_hash[:, k] = (prefix_hash[:, k - 1] * base +
                             solutions_array[:, k - 1]) % mod

    # Suffix hashes: suffix_hash[i, k] = hash of solutions_array[i, k:]
    suffix_hash = np.zeros((N, L + 1), dtype=np.int64)
    for k in range(L - 1, -1, -1):
        suffix_hash[:, k] = (solutions_array[:, k] * p_pow[L - k - 1] +
                             suffix_hash[:, k + 1]) % mod

    # for each index, we have a list of pairs
    result_pairs = [list() for _ in range(L)]

    # For each position, bucket by hash of vector with that position removed
    for pos in range(L):
        buckets = defaultdict(list)

        for idx in range(N):
            # prefix = hash of solutions_array[idx, :pos]
            prefix = prefix_hash[idx, pos]
            # suffix = hash of solutions_array[idx, pos+1:]
            suffix = suffix_hash[idx, pos + 1]
            # combine, shifting suffix down by one position in power terms
            combined_hash = (prefix * p_pow[L - pos - 1] + suffix) % mod
            buckets[combined_hash].append(idx)

        # Check pairs within buckets
        for idxs in buckets.values():
            if len(idxs) > 1:
                for i in range(len(idxs)):
                    index_sol_left = idxs[i]
                    val_left = solutions_array[index_sol_left, pos]
                    for j in range(i + 1, len(idxs)):
                        index_sol_right = idxs[j]
                        val_right = solutions_array[index_sol_right, pos]

                        if (val_left != val_right):
                            result_pairs[pos].append(
                                (index_sol_left, index_sol_right)
                            )

    return result_pairs

```

The complexity of this algorithm is harder to calculate, but anecdotally it is fast enough to be unnoticeable in normal use cases. The main parts of the algorithm iterate over each solution and each of its variables, and therefore its complexity should be  $L \times N$ . The individual operations to calculate the hash values are somewhat expensive to calculate, but the *scalability* of the overall algorithm is definitely better than the naive approach.

# C

## APPENDIX C: DEFINITIONS OF LINKAGE

This chapter is an addendum to Chapter 7.

First, it starts by setting some notation, and then it gives a comprehensive description and definition of each linkage method discussed in Chapter 7.

### C.1 NOTATION FOR LINKAGE

Pure mathematical notation is often hard to parse in the context of linkage learning. In the trade-off between mathematical purity and readability, this chapter leans more towards the latter.

The notation used for this chapter will mainly relate to solutions, **PSs** and their modifications. A solution may be indicated using a lowercase letter, usually  $x, y$  or  $s$ , and the apex  $\psi$  indicates a **PS**.

These can be modified by changing one variable:  $x_{a=0}$  represents the solution  $x$  but with the variable  $a$  set to 0. Similarly, two variables may be set as well:  $x_{a=0, b=0}$  is the solution  $x$  but with variables  $a$  and  $b$  set to 0.

The same rules apply to **PSs**, such as  $x_{a=*}^\psi$  being the pattern  $x^\psi$  but the variable  $a$  is set to  $(*)$ .

It is also sometimes necessary to modify all the variables *except* one.  $x_{a=*}^\psi$  is the pattern  $x^\psi$  but all the variables except for  $a$  have been set to  $(*)$ .

Extending from this notation,  $F(\cdot_{a=1})$  represents the fitness values of all solutions where  $a = 1$ .

Finally, linkage involves the definition of many functions, and in order to differentiate them it is necessary to give them long names. This sometimes involves giving them a name that is longer than one line, such as  $\text{first\_line}_{\text{second\_line}}(x)$ .

### C.2 LINKAGE LEARNING METHODS

This section will contain definitions of the **linkage** function, which takes two variables and returns how strongly linked they are. This is different from **atomicity**, which takes a **PS** and returns the linkage strength between all of its fixed variables. In other words, **atomicity** is built on top of the **linkage** function.

There are many kinds of linkage learning methods, and it is important to understand what makes them different.

#### c.2.1 The *F-link* example function

From now on, an example function will be used. Consider a function  $F_{\text{link}}$  with 3 binary combinatorial arguments, such that:

$$F_{\text{link}}(a, b, c) = \begin{cases} 0 \cdot (c + 2) & \text{if } (a, b) = (0, 0) \\ 1 \cdot (c + 2) & \text{if } (a, b) = (0, 1) \\ 5 \cdot (c + 2) & \text{if } (a, b) = (1, 0) \\ 4 \cdot (c + 2) & \text{if } (a, b) = (1, 1) \end{cases} \quad (\text{C.1})$$

An easier way to think about this function is:  $c + 2 \cdot (a \cdot 5 \pm 1)$ , where the  $\pm$  is determined by the combination of  $a$  and  $b$ .

This function is constructed such that:

- the *best* value for the variable  $a$  is always the same – it is always beneficial to have  $a = 1$ ,
- the *best* value for variable  $b$  is dependent on  $a$ ,  $b$  should have the opposite value to  $a$ ,
- the variable  $c$  is not linked to any variable, since its role in the formula is not affected by the others. Still, the final fitness value is entirely dependent on  $c$ , so perhaps every variable is dependent on  $c$ .

This function will be used to show faults in existing methods and current understanding of linkage. It introduces conditional behaviour, it is not linear and proposes a case where linkage is a matter of opinion.

### C.3 ISSUES WITH NON-LINEARITY DETECTION METHODS AND THEIR ISSUES

Many approaches to linkage detection can be described as **non-linearity** detection methods. More specifically, it is assumed that two variables are *not linked* if they have univariate effects on fitness which are **added together**.

More formally, it means that a fitness function where no linkage takes place is assumed to be in the form  $F(s) = F_1(s_1) + F_2(s_2) + \dots$ , which is an **univariate additively separable function**.

In many cases, this is a harmful assumption: there are many systems in which variables could be considered non-linked despite not fitting the above scheme. For example, a function could be a product of many univariate functions:  $F(s) = F_1(s_1) \cdot F_2(s_2) \cdot \dots$

In the coming paragraphs, issues relating to non-linearity detection methods will be discussed in detail, as these are the most commonly found in the literature.

Note that in the context of metaheuristics, the optimisation problem at hand is unlikely to be linear: if it was, metaheuristics would not need to be used.

#### c.3.1 *Inability to quantify linkage*

Let us consider a non-linearity detection method which produces a “0” when two variables are not linked. For any other value, there is no clear interpretation: it simply means that the variables are not linearly independent. It might be a coefficient in a bi-variate linear model, but even then it is an attempt to explain non-linearity using linear models, which is deceptive.

#### c.3.2 *The effect of non-linearity*

The assumption of additivity and linearity has effects which are best understood through a **mental experiment**: consider any optimisation problem, which has a certain fitness function. Upon applying a non-linearity detection method, linkage information is obtained in the form of a pair-wise linkage table.

Now consider the same problem, but the fitness function is fed **into a sigmoid function** (which is monotonic and reversible,  $\sigma(x) = 1 + \frac{1}{1+\exp(-x)}$ ): the optimal solution remains the same, and the ranking of each solution remains identical. In fact, a metaheuristic’s search process should look identical since the relationships between solutions remain the same.

Now, should there be a difference in linkage information between these two instances? This is a matter of opinion, but it should be a property that the user is aware of, since it does affect the results.

This effect can be observed in practice with ANOVA-based detection of linkage in Figure C.1.

For situations where this is an issue, this chapter proposes **dominance** based linkage. This method is robust against non-linearity, and it is designed to be more generally applicable due to its few assumptions.

In the coming sections, we discuss how Perturbation-based linkage, Analysis of Variance (ANOVA), SHAP, LIME and methods based on averages are non-linearity detection methods and share the aforementioned issues.

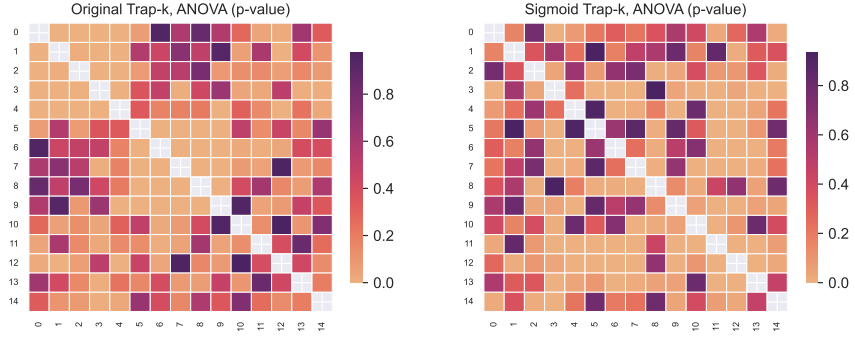


Figure C.1: A Trap-k problem (3 cliques of 5 bits) was solved by a standard GA, and the resulting set of solutions is analysed for linkage using ANOVA. On the left, the standard fitness function is used, whereas on the right the fitness function is the sigmoid ( $\sigma(x) = 1 + \frac{1}{1+\exp(x)}$ ) of the original fitness ( $F'(x) = \sigma(F(x))$ ). The sets of solutions are perfectly identical, and the only difference is in their fitness.

#### C.4 PERTURBATION-BASED LINKAGE

**Perturbation-based linkage** is perhaps the most traditional way of detecting an “interaction” between variables. A more accurate name for it would be **detection of non-linearity through perturbation**, as used by [202, 186]. It is a paired-samples, change-of-magnitude, effect-size, symmetric approach to linkage learning. Here, “perturbation” is used to mean a small change to a solution in order to observe its change in fitness.

Let us consider the simplest possible case, with a combinatorial binary problem. Then the perturbation-based linkage for a single solution is:

$$\text{perturbation\_linkage}_{\text{in\_solution}}(x, a, b) = F(x_{a=1}^{b=1}) - F(x_{a=0}^{b=1}) \quad (\text{C.2})$$

$$- F(x_{a=1}^{b=0}) + F(x_{a=0}^{b=0}) \quad (\text{C.3})$$

This approach uses new fitness evaluations and produces linkage data relating to a single solution. This formula is obtained by first measuring the **linear contribution** of each individual perturbation, then comparing it against the linear contribution of their combinations:

$$\text{lin\_cont\_*1} = F(x_{a=0}^{b=1}) - F(x_{a=0}^{b=0}) \quad (\text{C.4})$$

$$\text{lin\_cont\_1*} = F(x_{a=1}^{b=0}) - F(x_{a=0}^{b=0}) \quad (\text{C.5})$$

$$\text{lin\_cont\_11} = F(x_{a=1}^{b=1}) - F(x_{a=0}^{b=0}) \quad (\text{C.6})$$

$$\text{perturbation\_linkage} = \text{lin\_cont\_11} - \quad (\text{C.7})$$

$$(\text{lin\_cont\_*1} + \text{lin\_cont\_1*}) \quad (\text{C.8})$$

$$(\text{C.9})$$

Note that to calculate the full linkage information of a single binary solution, there will be  $\frac{L^2-L}{2}$  solutions to evaluate (one for every distinct pair of variables), which is already above 100 for 15 variables. Therefore, finding this linkage information even for a single solution is a rather expensive process, which is the main issue with this method. (The situation is even worse for non-binary combinatorial spaces, which are relevant to the Staff Rostering problem.)

Another important issue is that this method can only show when two variables are **not-linearly related** to the fitness function, which leads to many issues Section C.3.

##### c.4.1 Perturbation linkage definition

A full algorithmic definition of perturbation linkage is provided. The following elements need to be considered:

- Each combinatorial variable has a finite quantity of possible values, and in most systems these are encoded as integers starting at 0. The array `options` contains the **cardinality** for each of these, so that `options[a] == 2` indicates that  $a$  is a binary variable.
- To determine the “effect” of a variable  $a$  on a solution, we check the fitness change when changing the value of  $a$ . For example, setting  $a$  to 1 might increase the fitness by 2. For each possible value of  $a$ , we will keep track of how it affects fitness on average.
- To determine the “effect” of a pair of variables, we do a similar process but for every possible combination of values.
- The expected “effect” of two independent variables is that setting  $(a, b)$  to  $(v_1, v_2)$  is the effect of  $a = v_1$  + the effect of  $b = v_2$ .
- To get average effects, it is easier to sum them and then divide them at the end.
- The measured distance between the expected and observed effect of two variables is simply the sum of absolute value differences.

The algorithm definition is provided in Python, since the language of pseudocode would actually be more terse.

```
def perturbation_linkage(a, b, Pref, F, options):
    """ a, b are variable indices
        Pref is a set of solutions, with their fitness
        F is the fitness function, required for this
        options[a] is the cardinality of a
    """

    effect_a = np.zeros(shape=options[a])
    effect_b = np.zeros(shape=options[b])
    effect_ab = np.zeros(shape=(options[a], options[b]))

    for solution, fitness in Pref.items():
        for new_a in range(options[a]):
            new_solution = solution.copy()
            new_solution[a] = new_a
            new_fitness = F(new_solution)
            effect_a[new_a] += new_fitness - fitness
        for new_b in range(options[b]):
            new_solution = solution.copy()
            new_solution[b] = new_b
            new_fitness = F(new_solution)
            effect_b[new_b] += new_fitness - fitness

        for new_a in range(options[a]):
            for new_b in range(options[b]):
                new_solution = solution.copy()
                new_solution[a] = new_a
                new_solution[b] = new_b
                new_fitness = F(new_solution)
                effect_ab[new_a, new_b] += new_fitness - fitness

    effect_a /= len(Pref)
    effect_b /= len(Pref)
    effect_ab /= len(Pref)
    expected_ab = effect_a + effect_b.reshape(-1, 1)

    linkage_strength = np.sum(np.abs(effect_ab - expected_ab))

    return linkage_strength
```

#### c.4.2 Walsh decomposition

A generalisation of the above method is found in using **Walsh decompositions** of binary functions [127]. Every function that maps binary strings to scalars can be rewritten as a Walsh function [107], its Walsh decomposition. This formulation, and how this decomposition can be obtained is defined in [128].

In summary, the fitness function is rewritten as the sum of many terms, where the coefficients of certain terms indicate linkage strength. The main benefit of this approach is that it extends beyond pairwise relationships, but at the same time it is restricted to binary problems.

For pairwise interactions, perturbation and Walsh coefficients are mathematically equivalent [127].

#### c.5 FITTING LINEAR MODELS AND ANOVA

Many linkage learning methods are extensions of variable importance methods. A common variable importance method is fitting a linear model in order to approximate a function, and then inspecting its coefficients. Therefore, it might seem obvious to apply the same process for linkage learning: simply use a model with **interaction terms**, and check their coefficients.

An ANOVA test for interaction works very similarly, but instead it measures the *probability* that the coefficient for the interaction term is 0 [8, 160].

##### c.5.1 Building a naive linear model with interactions

First, let us consider a linear model which will attempt to model  $F_{\text{link}}$ , which takes as input 3 binary values  $(a, b, c)$ . The linear model will have the following parameters to be learned:

- intercept
- coefficients for  $a, b, c$
- coefficients for pairwise **interaction** terms  $(ab, ac, bc)$

Similarly to how a linear model can be used to obtain variable importance by looking at its coefficients, this model has “pairwise” terms with coefficients that quantify linkage (e.g. the coefficient for  $ab$  is the linkage strength between  $a$  and  $b$ ).

If  $F_{\text{link}}$  was a perfectly linear function ( $k_0 + k_1a + k_2b + k_3c$ ) then all the coefficients for the interaction terms would be 0. If  $F_{\text{link}}$  is not a linear function, then the value of those coefficients is unpredictable and yields poor results (as it is **underfitting**, a linear model can only model linear functions).

Let us try to fit a linear model to this function:

$$F(a, b, c) = (a + 1) \cdot (b + 1) \cdot (c + 1)$$

We will look at the coefficients to determine if there is linkage between the variables. Whether these variables are linked or not is up for debate, but it is sensible to expect that they are all **equally** linked to each other. Let us fit the aforementioned linear model, and the following is obtained:

$$1.375 + 0.25a + 0.25b + 0.25c + 2.5ab + 0.5bc + 2.5ac$$

This linear model has  $R^2 = 0.96$ , which indicates that it is highly accurate. Despite this, the pairwise coefficients are 2.5, 0.5 and 2.5, which does not reflect our expectation of equal linkage.

##### c.5.2 Building a more sensible linear model

The linear model shown in the previous section has issues which are commonly seen in the literature.

**TOO MANY TERMS** The model shown above contained all the univariate terms, and all the pairwise interaction terms. This approach quickly becomes unsuitable as more input variables are present (for stability and computational complexity), with 14 variables requiring a linear model with 106 terms already. A more scalable approach is to make *a linear model for every pair of variables*, so that the linear models are always in the following form:

$$\text{Fitness} \sim A + B + A : B \quad (\text{C.10})$$

The drawback is that instead of one big model, there are many small models, but at least these will be stable.

**EXACT MEANING OF THE INTERACTION TERM** Consider a case where the variable  $a$  has possible values {Red, Green, Blue}, and a variable  $b$  has possible values {White, Black}. The term  $a : b$  in the linear model is actually 6 terms, one for each possible combination of their values. For example, there is a term  $(A == \text{Red}) \cdot (B == \text{Black})$  which has value 1 only when  $A$  is Red and  $B$  is Black.

The interaction term used in linear models is different between **combinatorial** variables and **continuous** variables, where it is simply a multiplication of values. The models shown in Section C.5.1 intentionally showed the wrong kind of interaction (multiplication of values), in order to show how easy it is to get confused. For binary variables, each pair of variables should result in 4 terms in a model:  $k_1ab + k_2(a-1)b + k_3a(b-1) + k_4(a-1)(b-1)$ .

This aspect is strongly connected with the point made in Section C.5.2: making a linear model with all the interaction terms would be far too large, and it is necessary to make many small models instead. Another point to note is that the meaning of this interaction term is only well defined for additive systems. A paper from 1948 relating to ANOVA states [90]:

When the interaction is not negligible, estimation of the main effect of  $A$  must depend upon circumstances, and no definition [...] can be claimed as ‘the only legitimate’.

In general, the nature of this interaction, and how this should be analysed is a matter of contention [131].

**OBTAINING THE COEFFICIENTS** ANOVA models implicitly use the aforementioned models, and Ordinary Least Squares is the most common method of obtaining the coefficients [144]. There is one detail that is often overlooked for ANOVA: the **order** in which the coefficients and p-values are obtained [131, 169, 63]:

- Type I: the coefficients are obtained sequentially. The  $(n+1)$ th term is based on the residuals from fitting the previous  $n$  terms.
- Type II: the coefficients are obtained in isolation, and they are not influenced by each other’s residuals.
- Type III: the coefficients are obtained all at the same time.

Of these, type II is the most suitable for obtaining the coefficients to the interaction terms, as it removes any unnecessary influence from other variables and is the most robust in the case of unbalanced data [169].

**P-VALUE OR EFFECT SIZE** An ANOVA test produces many numerical results, with two being the most important: the p-value and the effect size. As already discussed in Section 7.2.4, these represent two different measurements. If one wants to determine *if* there is linkage, they should use the p-value, while they should use the effect size (called  $\eta^2$ ) to quantify its strength.

### c.5.3 ANOVA linkage definition

In summary, the suggested definition of linkage through ANOVA for a pair of variables  $(a, b)$  is:

- For every solution in the dataset, store the value of  $a$ ,  $b$  and its fitness.

- Define a linear model in the form  $\text{Fitness} \sim a + b + a : b$
- Obtain the linear model through Ordinary least squares, then carry out Type II ANOVA
- Extract the p-value or effect size of the statistical test.

Once again, this algorithm lends itself better to Python code than to pseudocode.

```
def ANOVA_linkage(a, b, Pref):
    """
    Calculates the interaction between variable A and variable B
    using a Two-Way ANOVA (Type II).
    solutions = Pref.solutions # a matrix where each row is a solution
    fitness = Pref.fitness # an array of all the fitness values
    """

    # Construct a DataFrame
    df = pd.DataFrame({
        'A': pd.Categorical(solutions[:, a]),
        'B': pd.Categorical(solutions[:, b]),
        'Fitness': fitness
    })

    # Check if enough samples are present
    if df['A'].nunique() < 2 or df['B'].nunique() < 2:
        return 1.0

    # Define the linear model
    model = ols('Fitness ~ C(A) + C(B) + C(A):C(B)', data=df).fit()

    # Perform ANOVA (Type II)
    anova_table = sm.stats.anova_lm(model, typ=2)

    # Extract p-value.
    p_val = anova_table.loc['C(A):C(B)', 'PR(>F)']
    # If we wanted effect size, use 'sum_sq' instead of 'PR(>F)'

    return p_val
```

## C.6 SHAP INTERACTION VIA XGBOOST

SHAP is a well-known method for measuring variable importance, and therefore there is a method for obtaining linkage using similar techniques, namely XGBoost [50].

An introduction to Shapley values was given in Section 3.5.2, and how SHAP was derived from it is provided in Section 3.5.2.

What the reader needs to be aware of for this section is that Shapley assumes that the underlying function is additive, and SHAP methods make even stronger assumptions in order to be computationally feasible. Moreover, these methods rely on the presence of a model, which adds a layer of indirection (is the linkage within the model or within the data?). At the same time, the use of the model might allow for SHAP methods to detect non-linear effects.

### c.6.1 SHAP linkage definition

The SHAP linkage learning proposed here will be based on an XGBoost model[50]. This is chosen as it is the most popularly used method of linkage via SHAP. It is implemented as follows by `scikit-learn`:

- The dataset of solutions and fitness values is used to train a surrogate model, specifically an XGBoost model.
- Within the model, interaction terms based on SHAP are calculated. Here, a SHAP value is the contribution of a *pair* of variables.
- SHAP values explain individual solutions. To obtain general values, these are averaged over all solutions

In simple terms, XGBoost consists of an ensemble of shallow decision trees, and two variables are interacting if they tend to be used within the same trees.

```
import xgboost as xgb
import shap

def SHAP_linkage(a, b, solutions, fitness):

    # Train the surrogate model
    model = xgb.XGBRegressor(n_estimators=100, max_depth=6)
    model.fit(solutions, fitness)

    # Calculate SHAP interaction values using TreeExplainer
    explainer = shap.TreeExplainer(model)
    interaction_values = explainer.shap_interaction_values(solutions)

    # get averages over solutions
    global_interactions = np.abs(interaction_values).mean(axis=0)

    return global_interactions[a, b]
```

**TIME COMPLEXITY** An issue with this method is its scalability: for this linkage information to be obtained, it needs to train a model, potentially on many variables. This becomes increasingly slow as the quantity of variables and solutions increases. If certain parameters are kept constant (quantity of trees, depth of trees),  $L$  is the quantity of variables and  $N$  is the quantity of solutions, then the time complexity of model training is  $\mathcal{O}(LN^2 \log N)$  [50]. The time complexity of calculating the linkage values is not known, but it is expected to be insignificant with regard to the model training step. This scales poorly as more solutions are to be considered, and it is expected to become unfeasible for highly-dimensional problems.

## C.7 CORRELATION

The concept of correlation belongs to the conditional-distribution category of methods, generally for continuous variables but not necessarily.

The most popular methods for continuous variables are the following.

- **Pearson Correlation coefficient ( $r$ )**: assumes that between two variables there is a linear relationship with normally distributed noise. This method produces a value between -1 and +1 that represents the strength and direction of the estimated linear relationship [13].
- **Spearman's Rank Correlation ( $\rho$ ), Kendall's  $\tau$** : these measure if there is a monotonic relationship between variables, by comparing the rankings of their values. These do not assume a linear relationship or normality of noise, which makes them non-parametric alternatives to Pearson correlation [13].
- **Chatterjee's  $\xi$  correlation**: an asymmetric metric which measures if a variable can be predicted by knowing another. More formally, it determines whether there is a function that can map between variables. More specifically, it measures how discontinuous the relationship between rankings is [48].

For combinatorial variables, the following are available:

- $\chi^2$  test for independence ( $\chi^2$ ): (discussed in Section C.8): measures the **probability** that two categorical variables are linked by a conditional distribution.
- $\phi$  coefficient, Cramer's V: (discussed in Section C.8) these measure the **strength** of two combinatorial variables.  $\phi$  is limited to two binary variables, whereas Cramer's V applies to higher cardinalities.

Finally, there are methods that work between a combinatorial and a continuous variable, such as **biserial** and **point-biserial** correlation [262].

As this thesis focuses on combinatorial problems, the correlation between continuous variables will not be considered.

## C.8 $\chi^2$ TESTS

Other than ANOVA, there are other statistical approaches that can be used for linkage learning, and these are based on conditional distributions.

Given a set of solutions, we can count the probability for each variable to have a certain value (the **marginal distribution**). Similarly, the **joint distributions** will show the probability of each *combination* of values for two variables.

If two variables are linked, we should observe a conditional probability distribution. For example, if the variable  $u = 1$  30% of the time, and the variable  $v = 1$  30% of the time, we would expect  $(u, v) = (1, 1)$  to occur 9% of the time if the variables are independent. If those variables are not independent, then the combination  $(1, 1)$  would have a higher or lower proportion than 9%. In the case of a metaheuristic, the combination  $(1, 1)$  might be more common than expected due to natural selection.

The important step is to *quantify* how strongly a given joint distribution “behaves as a conditional distribution”, and there are many statistical tests to achieve this. The most common is a  $\chi^2$  **test**, which calculates the probability that the observed joint distribution came from independent marginal distributions [238]. The result of this test is a p-value and an effect size (Cramer’s V).

$\chi^2$  tests are an approximation of Fisher’s Exact test. Fisher’s exact test is rather expensive and complex to implement, and most implementations only allow for binary variables.  $\chi^2$  tests produce nearly identical results, and are used throughout this thesis. There are many other derivations of Fisher’s test, such as Boschloo’s Test [30], G-test [189, 66], Phi-coefficient [65], and the Fowlkes-Mallows index [93]. These aim to correct certain biases in different ways.

These kinds of tests disregard fitness values, placing them in the **conditional-distribution** category of linkage learning methods.

### C.8.1 $\chi^2$ linkage

$\chi^2$  tests are the most well-documented in the literature, with definitions being available in [38] for example.

To quantify linkage between two variables A and B, the following is done:

1. Create a zero matrix of dimensions  $|D_A| \times |D_B|$ , called **counts**.
2. Iterate through every solution  $s$  in the  $P_{\text{Ref}}$ . Increment **counts** $[s_A, s_B]$ .
3. **counts** now represents the conditional distribution between A and B. This can be passed as the input to a  $\chi^2$  test.
4. From the statistical test, extract either the p-value or the effect size. Cramer’s V is recommended due to its wide applicability.

## C.9 MUTUAL INFORMATION AND BAYESIAN LINKAGE

The following methods of detecting linkage have been popular in many EDAs, as they are flexible and can produce linkage information from a population of solutions. In information theory, this is measured via **Mutual Information**. If the probability of a variable having a certain value is a piece of information (think of the game “Guess Who?”), Mutual Information measures how much knowing the value of one variable predicts knowing the value of the other, as defined in Equation (C.11).

$$MI(a, b) = \sum_{\substack{\text{val}_a \in \text{options}(a) \\ \text{val}_b \in \text{options}(b)}} p(\cdot_{a=\text{val}_a} \cdot_{b=\text{val}_b}) \log \left( \frac{p(\cdot_{a=\text{val}_a} \cdot_{b=\text{val}_b})}{p(\cdot_{b=\text{val}_b})p(\cdot_{a=\text{val}_a})} \right) \quad (\text{C.11})$$

(where  $p$  indicates the probability of a solution having certain values)

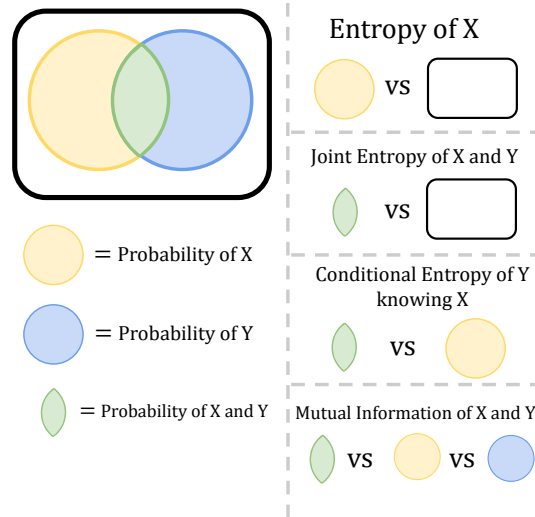


Figure C.2: Diagram showing the elements involved in Joint Entropy, Conditional Entropy and Mutual Information. Mathematical notation is intentionally avoided in order for the reader to focus on what elements are used in each formula.

The terms Mutual Information, Joint Entropy and Conditional Entropy are often confused.

**Entropy** is a function of probability,  $H(p) = -p \cdot \log(p)$ .

**Joint Entropy** is the entropy of the probability of two variables having a certain combination of values.

**Conditional Entropy** is the entropy of a probability of an event, knowing that another event has occurred. This is non-symmetric.

**Mutual Information** is not an entropy, and it compares 3 probabilities. A summary is shown in Figure C.2.

**Bayesian methods** use similar formulas, but they are hidden within a larger process. Bayesian methods aim to generate a **directed graph**, where the nodes are the variables of the problem (as shown in Figure C.3). A node points to another if it influences its distribution, and thus the graph can represent complex dependencies (one variable is dependent on two variables, which themselves depend on others...) but finding such a graph is not easy. Generally, many such graphs are generated, and the one which maximises a certain metric is picked (usually this is implemented as a greedy search) [216]. This metric to be optimised is often based on information theory.

In the Bayesian Optimisation Algorithm (BOA) family of methods for example, two metrics are used: the Bayesian Information Criterion and the Bayesian-Dirichlet metric with likelihood equivalence [216]. The first method uses Conditional Entropies, whereas the second is a distance between the observed distribution and the one generated by the graph [218, 216].

These methods fall in the conditional-distribution category, and they are implemented in Equation (C.11). Although there are many similar metrics, they are expected to perform similarly in our context since they cannot make use of fitness information.

#### C.10 FRIEDMAN'S H STATISTIC

Friedman's H statistic is isolated from other methods, and it has received relatively little attention in the literature. It is a way to quantify conditional-effect on fitness, and it stems from thinking about the *gradients* of the fitness function [95]. The theory behind this method is based on Partial Dependence plots, which in practice are surrogate models for the fitness function that use only certain variables. Therefore, its implementation is somewhat similar to Shapley values, but it has less layers of abstraction.

Firstly, a fitness function for a problem with two variables  $j$  and  $k$  could be written as  $F(x) = F_j(x_j) + F_k(x_k) + F_{jk}(x_j, x_k)$ , but if the variables are independent then it could

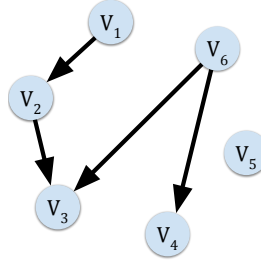


Figure C.3: Example of a Bayesian Network for an optimisation problem with 6 variables. Each variable is a node, and the arrows indicate conditional distributions. For example,  $V_2$ 's value distribution is conditional to  $V_1$ 's value.  $V_3$  is conditional to the combination of values of  $V_2$  and  $V_6$ . The distribution of  $V_5$  is independent of other variables.

be  $F(x) = F_j(x_j) + F_k(x_k)$ . This framework is the starting point of this methodology, and therefore this method is also based on non-linearity detection.

Note that  $F_j$  and similar terms are not estimates of fitness, but rather they are terms of a bigger sum. It might be easier to think of them as correction factors.

The final metric is derived as follows:

1. For a problem with  $L$  variables,  $F_l$  can be approximated by using certain models such as a random forest regressor where the input variables can be chosen. This approximation of the correction factors is represented by  $\hat{F}_j$  and similar terms. In the case of pairwise linkage for combinatorial variables, a model is not required and  $\hat{F}$  summands are simply averages.
2. Let  $\hat{F}_{jk}(x_j, x_k)$  be the predicted correction factor for the fitness of  $x$  based on a model that only uses variables  $j$  and  $k$ .
3. For a solution  $x$ , one can measure the “local linkage” for an individual solution  $x$  for  $j$  and  $k$  (`sol_variance`, Equation (C.12)).
4. The local linkages can be aggregated over all solutions, producing a value which can be described as the **variance of the interaction term** (`jk_variance`, Equation (C.13)).
5. There is variance of the overall model which uses  $j$  and  $k$  (with  $F_j$  and  $F_k$  correction factors). This is `model_variance`, in Equation (C.14).
6. The final linkage metric is `jk_variance` normalised by the `model_variance`, in Equation (C.15).

$$\text{sol\_variance}(x, j, k) = \hat{F}_{jk}(x_j, x_k) - \hat{F}_j(x_j) - \hat{F}_k(x_k) \quad (\text{C.12})$$

$$\text{jk\_variance}(j, k) = \sum_{x \in P_{\text{Ref}}} (\text{sol\_variance}(x, j, k))^2 \quad (\text{C.13})$$

$$\text{model\_variance}(x, j, k) = \sum_{x \in P_{\text{Ref}}} (\hat{F}_{jk}(x_j, x_k))^2 \quad (\text{C.14})$$

$$H_{jk}^2 = \text{jk\_variance}(j, k) / \text{model\_variance}(j, k) \quad (\text{C.15})$$

The definition provided is broken down in a way that is not suggested in the original formulation, but they are mathematically equivalent. This is for the benefit of the reader.

#### C.11 COSINE SIMILARITY LINKAGE

As part of the work with Recommender Systems ([43]), the linkage measure used made use of the fact that each variable was associated with an embedded vector (denoted with

`embedding(a)` for variable  $a$ ). Cosine similarity is a commonly used technique in machine learning, as it allows to define a similarity metric between items, aligned with domain information (Equation (C.16)).

$$\text{cos\_sim}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|} \quad (\text{C.16})$$

Linkage between two variables is trivially defined as:

$$\begin{array}{l} \text{linkage} \\ \text{cosine\_sim} \end{array} (a, b) = \text{cos\_sim}(\text{embedding}(a), \text{embedding}(b)) \quad (\text{C.17})$$

# D

## APPENDIX D: LINKAGE TABLES

This appendix will present a selection of linkage tables obtained by applying the various linkage learning algorithms described in Chapter 7. These are applied on the metaheuristics and benchmark problems described in the same chapter.

The linkage tables shown in Figure D.1 serve as a baseline for what the correct linkage tables should look like for every linkage problem. Note that the linkage table for a benchmark problem and its Sigmoid counterpart are identical.

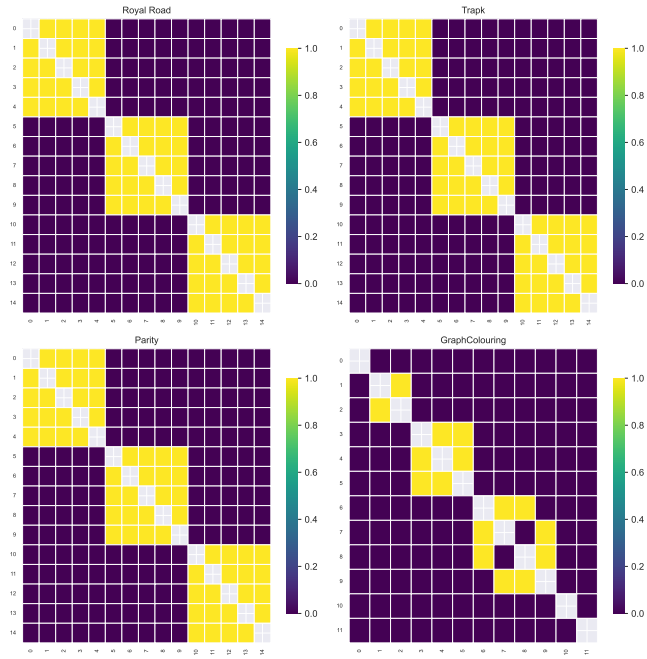
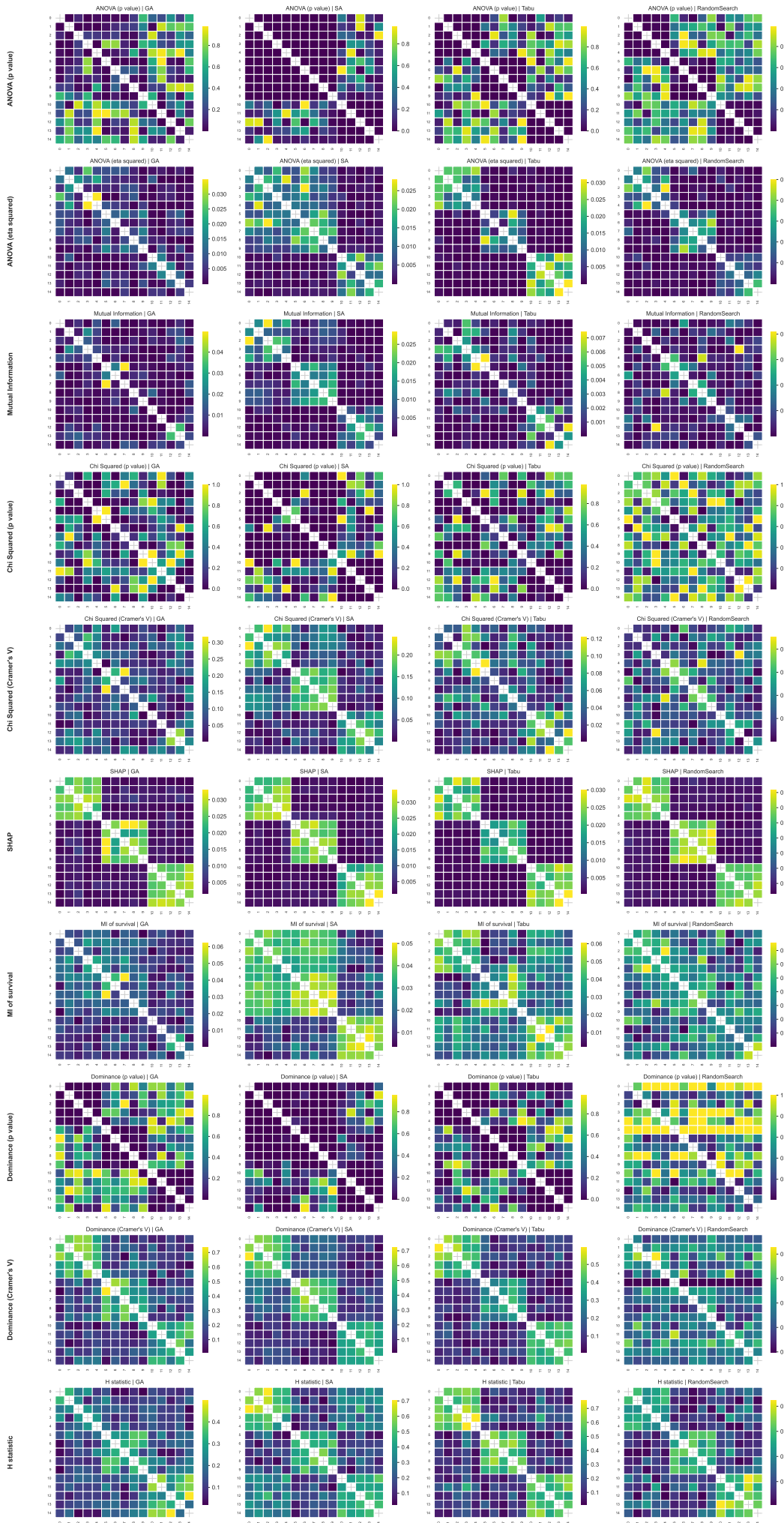
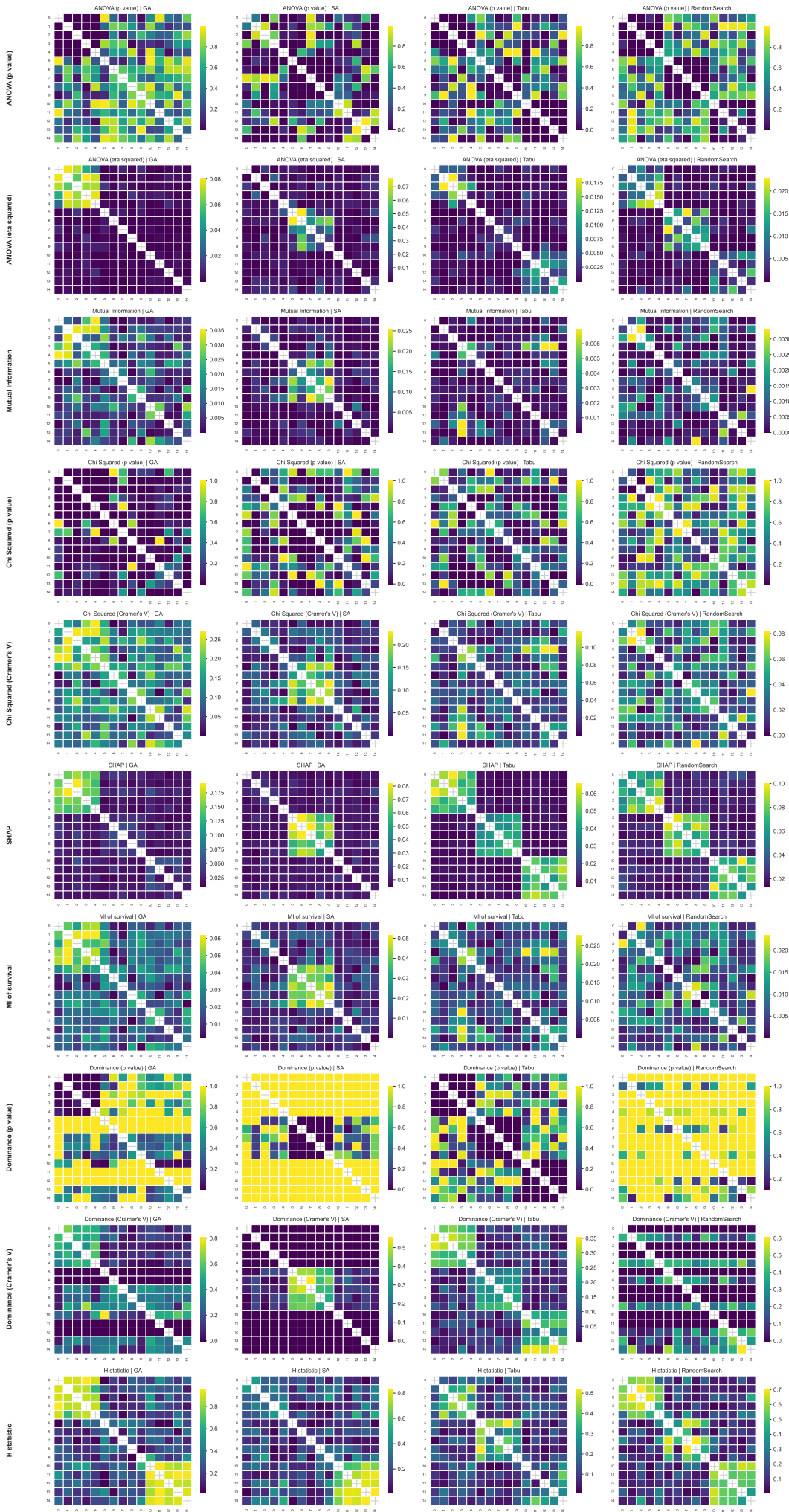


Figure D.1: Linkage tables representing the pairwise linkage relationships between the variables of each benchmark optimisation problem. Yellow cells mark pairs of variables which are linked.

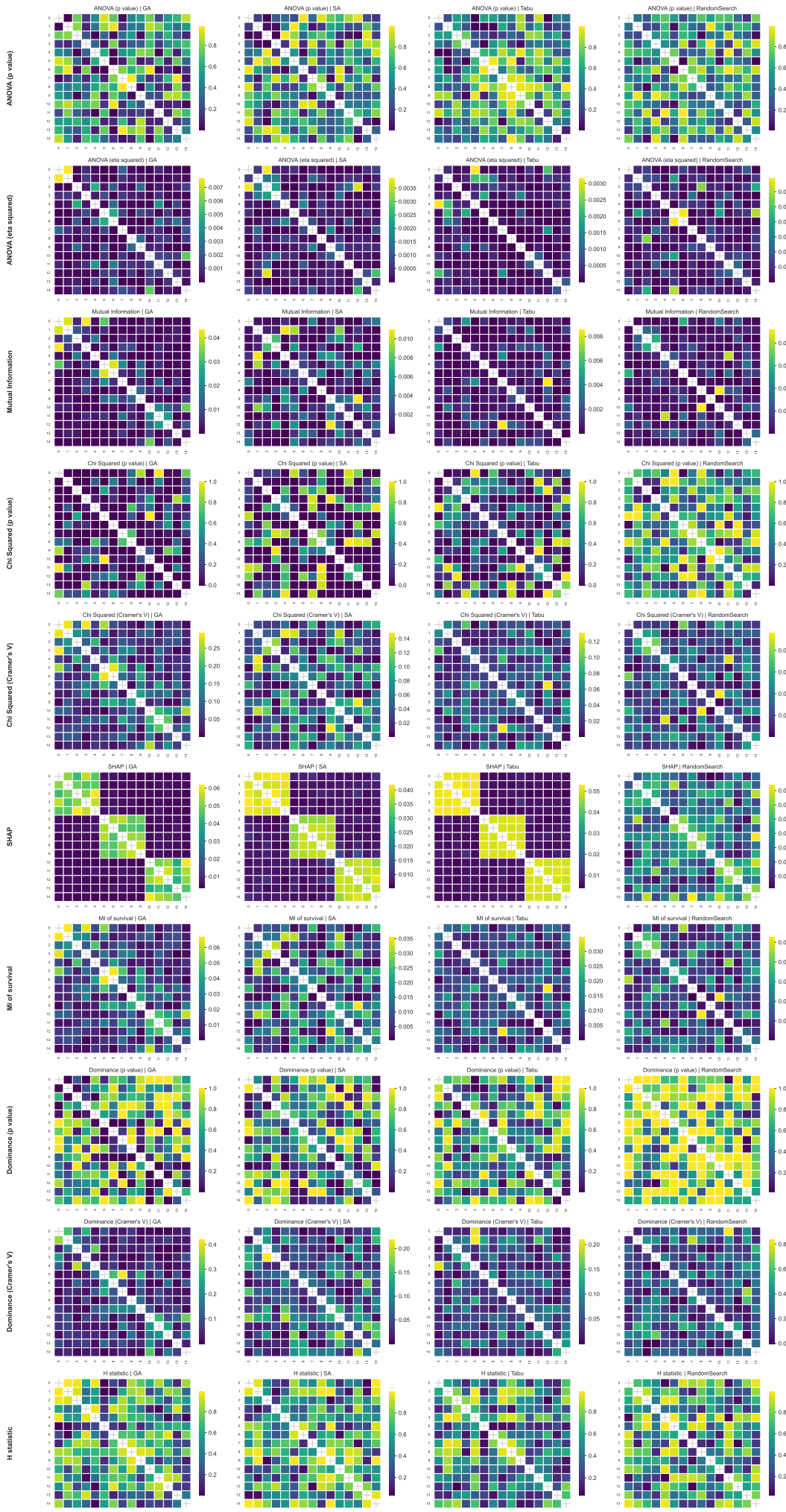
# Problem: Royal Road



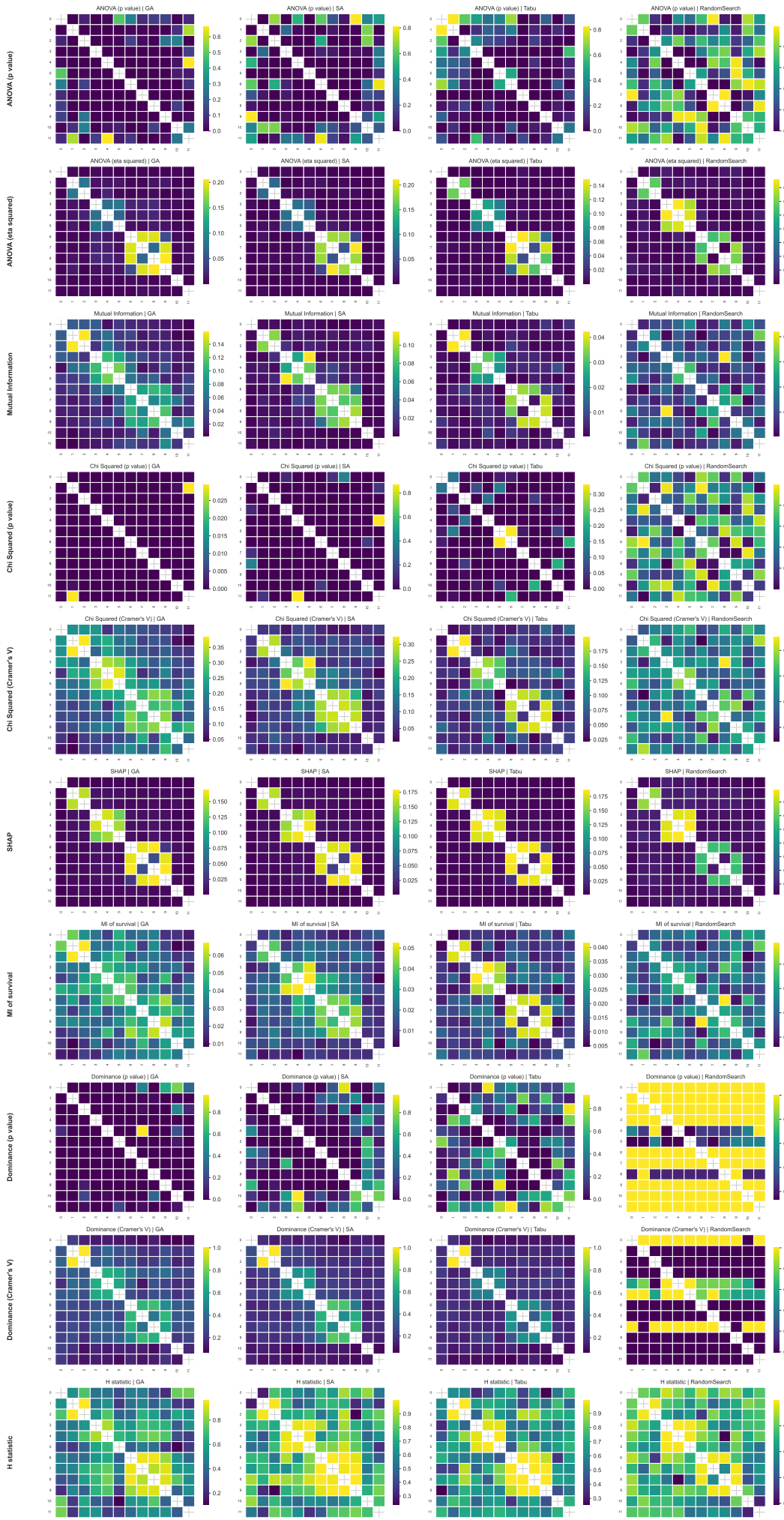
Problem: Trapk



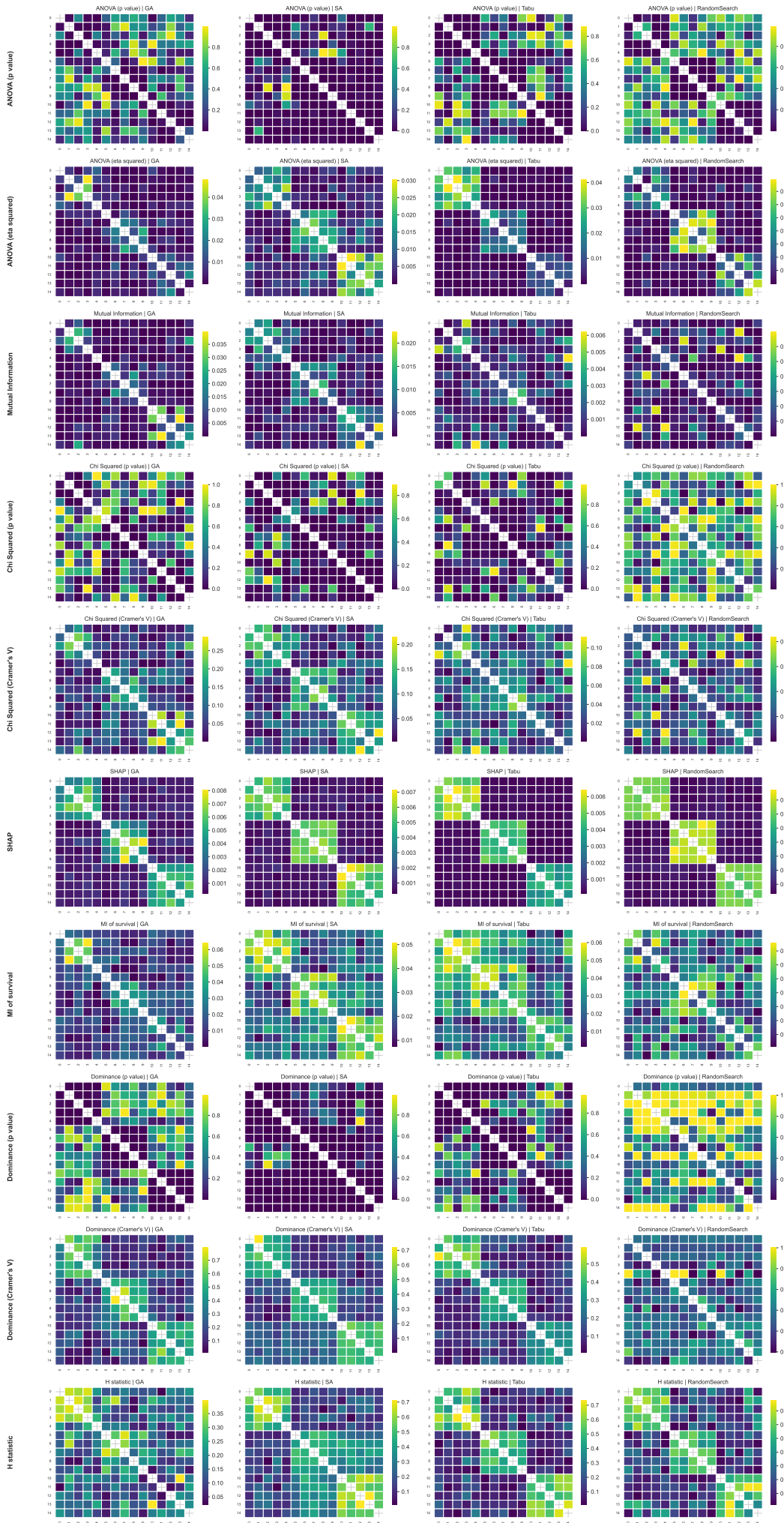
# Problem: Parity



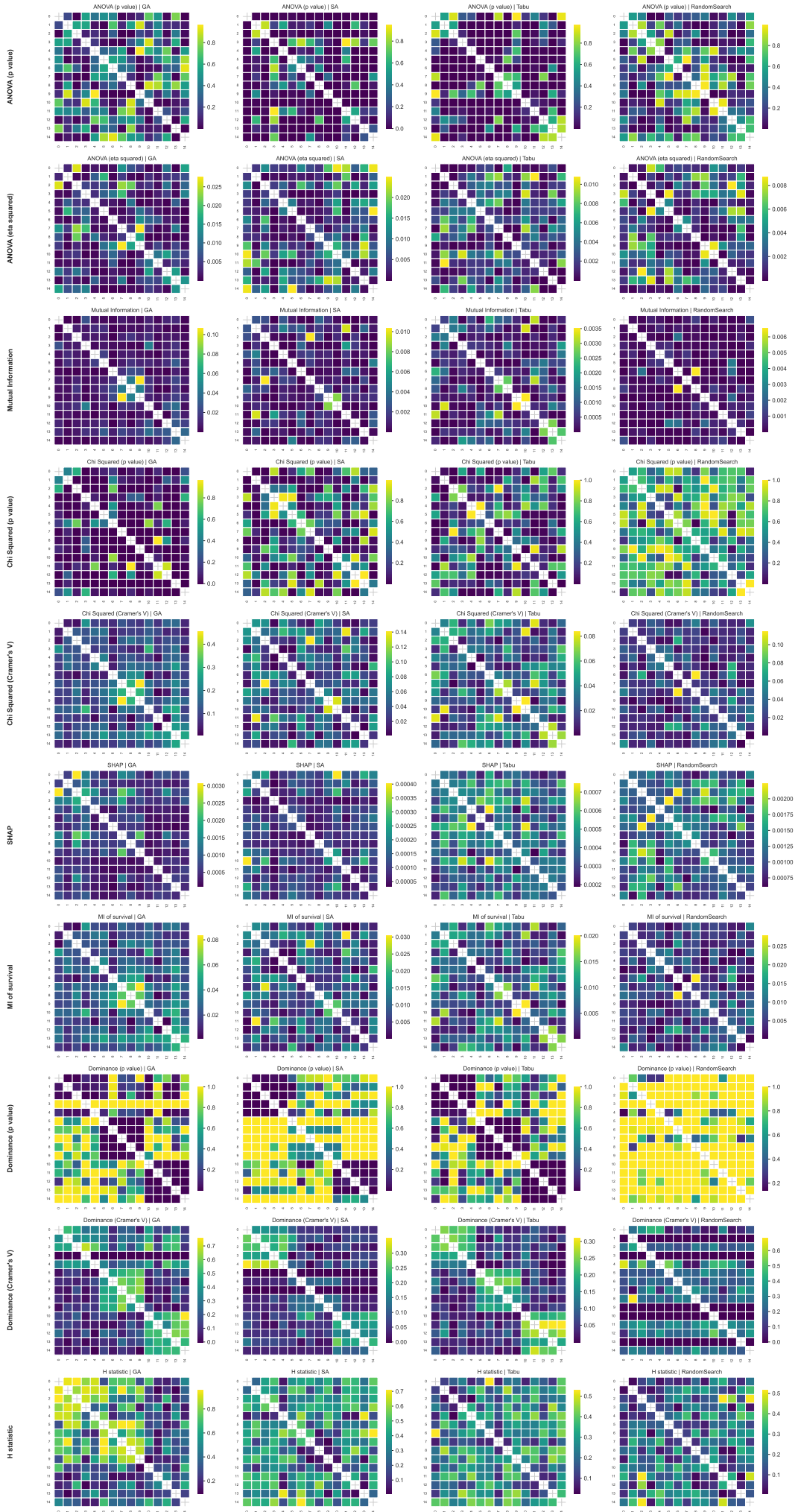
# Problem: GraphColouring



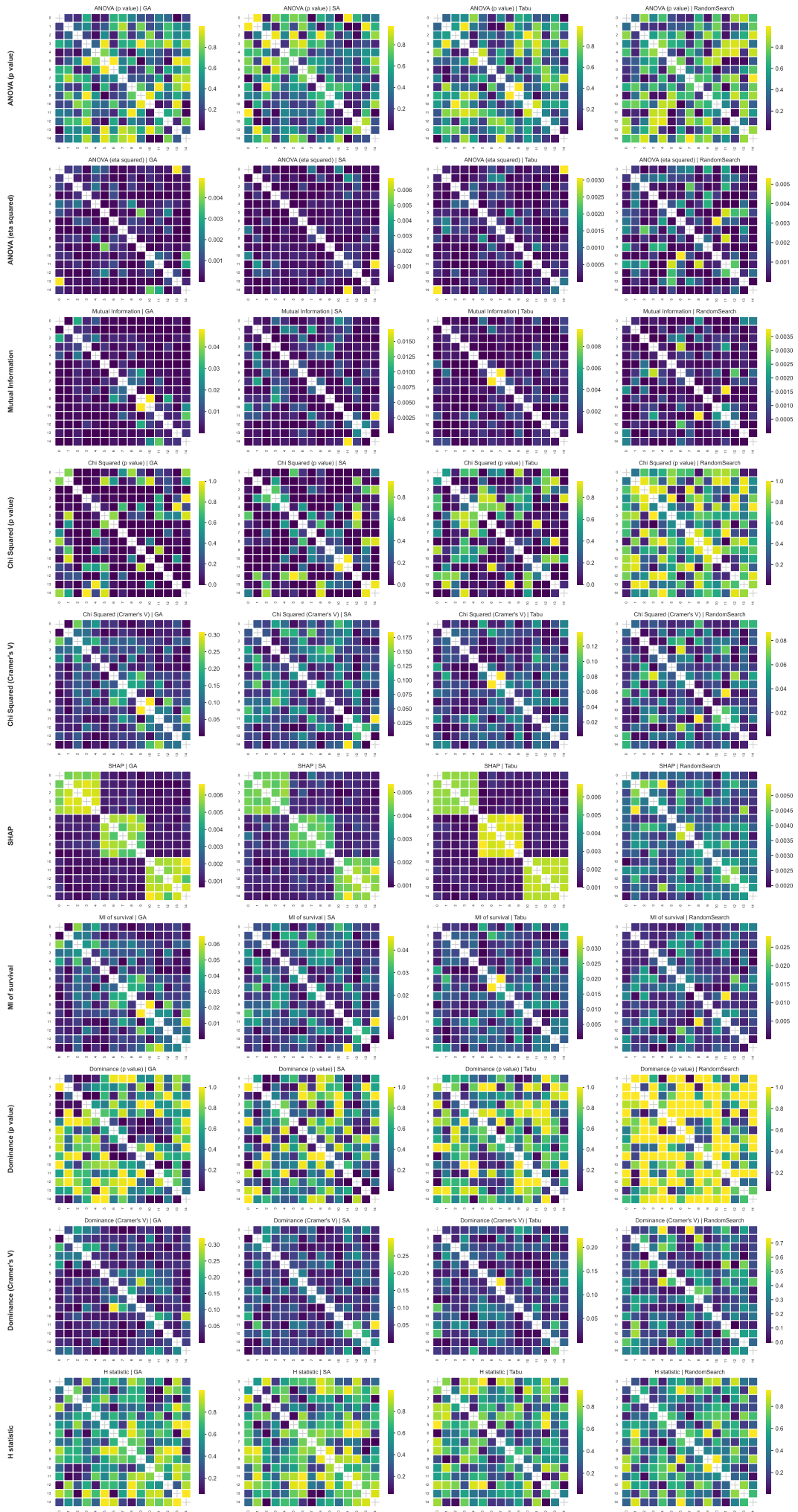
# Problem: Sigmoid Royal Road



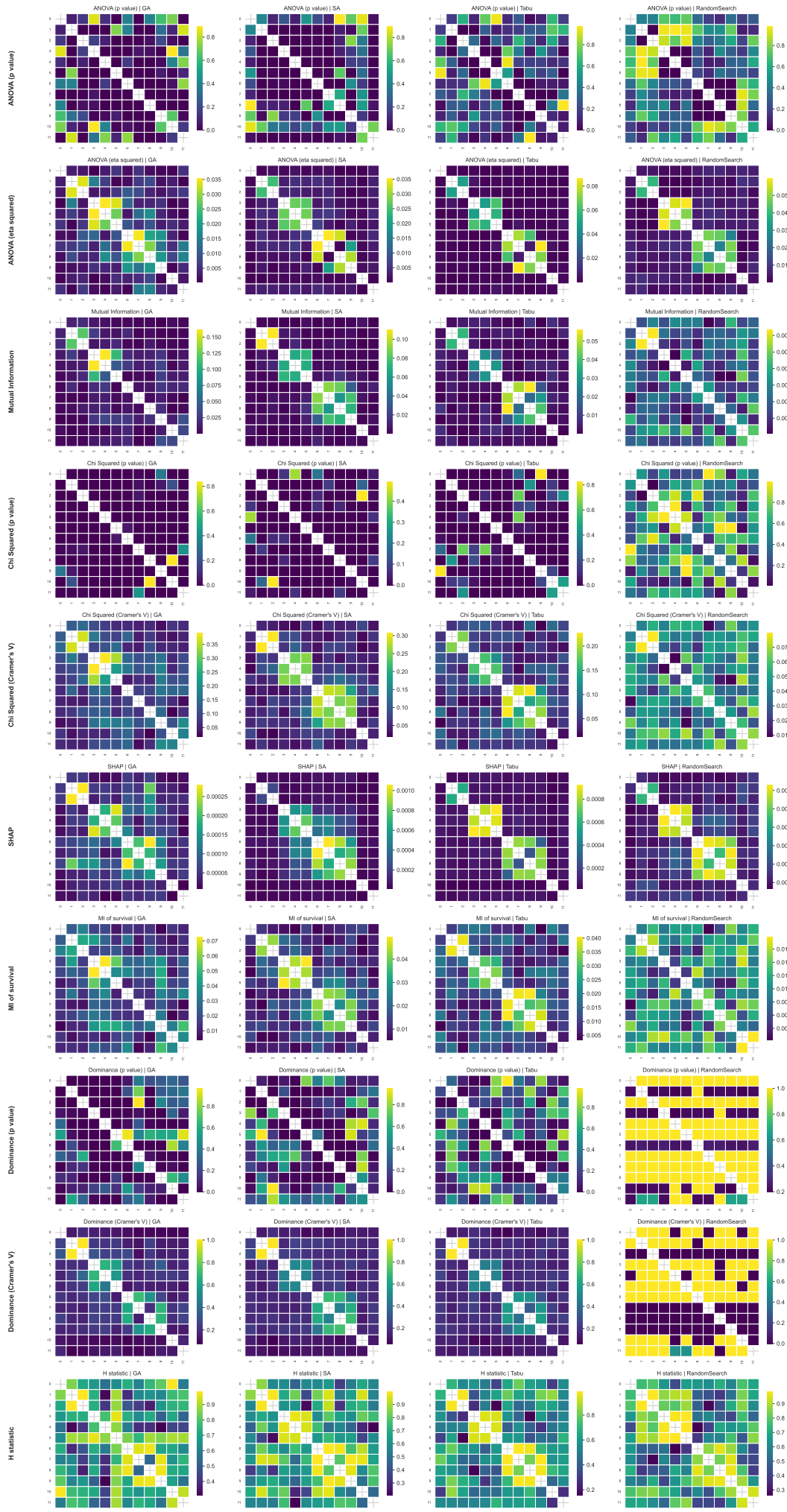
# Problem: Sigmoid Trap



# Problem: Sigmoid Parity



# Problem: Sigmoid GraphColouring



## APPENDIX E: BENCHMARK STAFF ROSTERING PROBLEMS

## E.1 DEFINITION OF SR-RR

This Staff Rostering problem instance is equivalent to a RoyalRoad problem where each clique contains 4 variables, and there are 5 cliques in total.

There are 20 workers in total, and 5 skills. Every clique in the original RoyalRoad problem corresponds to a group of 4 workers, each having the same skill. In the first clique, the workers have skill A, in the second they have skill B etc...

Each worker has the rotas shown in Figure E.1, each being 5 weeks long: one rota with no working days, and a rota where they work every day for one week. The first worker can work on the first week, the second worker can work on the second week etc...



Figure E.1: Rota choices available for

This group of 4 workers is repeated 5 times, with skills A, B, C and D respectively. Within the clique with skill A, if all the workers use rota #1 then every day is covered by exactly one person with skill A. If even one worker in the clique uses rota #0, then there is a day where 0 workers with skill A are working. In these latter cases, the minimum observed quantity of workers is 0, which causes the Range of that skill to be a low constant value (division by 0 is prevented by returning a low constant value).

Finally, the weight of each day is set to 1, and the weight of preferential rotas (W) is set to 0.

The final fitness is the summation of the Range for every skill, and so the final problem is equivalent to RoyalRoad.

## E.2 DEFINITION OF SR-GC

This instance of the Staff Rostering problem is equivalent to a Graph Colouring problem composed of isolated 3-node cliques, with 3 colours available, shown in Figure E.2.

For this problem, there are 15 workers and 15 skills in total (A to O). For each clique in the Graph colouring problem, there are 3 skills assigned. In the first clique, the first worker has skills A and B, the second has skills B and C, and the third has skills C and A. This is done so that the choice of rota for the first worker affects both of the other workers in the clique *independently* (because each pairing shares a skill, but they do not all share a skill). Each worker has 3 rota options, where in each rota the worker will

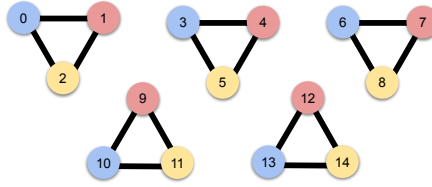


Figure E.2: Graph Colouring problem corresponding to SR-GC

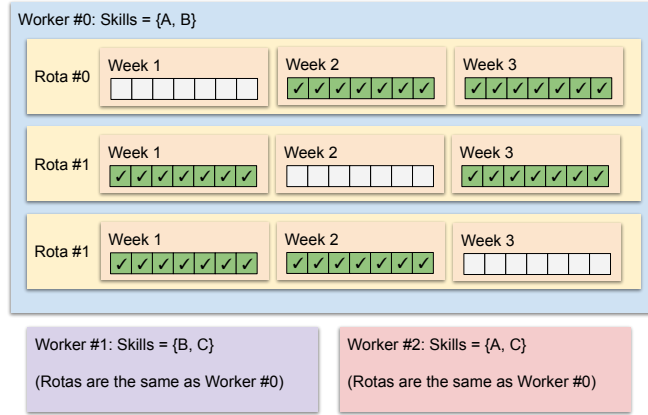


Figure E.3: Workers and their rotas corresponding to the first clique in the Graph Colouring problem.

work every day except for one week. This setup, shown in Figure E.3 is then used for every clique in the Graph Colouring problem.

Each clique's optimal rota assignment is when the workers choose different rotas from each other, just like in Graph Colouring. For a specific skill, if the two workers with that skill choose the same rota, there will be a week with zero workers, and thus the Range will have a high penalty. If those two workers choose different rotas, there will be a week with 2 people every day, and two weeks with one person every day, which has a better Range. In other words, each skill acts as an "edge" of the network, and if the two nodes have the same colour, then a penalty is applied.

This setup was verified using a computer program, where it was shown through enumeration that there is a monotonic and reversible relationship between the fitness values of SR-GC and the true Graph Colouring instance.

APPENDIX F: QUESTIONNAIRE FOR USER STUDY

---

The following are the questions contained in the questionnaire administered as part of the user study discussed in Chapter 9. There are 2 such questionnaires, and this is Variant 'A': there are explanations for the first task but not for the second. Variant 'B' lacked explanations for the first task, but had them for the second task.

There are various links shown, which at the time directed the user to documents such as the “tips and tricks” (shown in Figure 9.2) and explanations (shown in Figure 9.1).

About you

These are just simple questions about you, but the quiz will remain anonymous.

4. Are you a current student at the university of Stirling? \*

- ☐ Yes  
☐ No

5. If you are a student, what is your degree and in what year are you in?  
(E.g. "Undergraduate 3rd Year Computing Science")

6. If you are not a student, how would you describe yourself in terms of **occupation**? (might be chef, homemaker, unemployed...)

7. How would you rate yourself in the following technical skills? \*

|                                                                                                                            | Basic                 | Limited               | Intermediate          | Proficient            | Expert                |
|----------------------------------------------------------------------------------------------------------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| <b>Numeracy</b> (<br>working with<br>numbers, basic<br>arithmetic, and<br>mathematical<br>problems)                        | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| <b>Problem<br/>Solving</b> (<br>tackling<br>technical<br>challenges and<br>apply creative<br>solutions)                    | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| <b>Algorithmic<br/>Thinking</b><br>(Ability to break<br>down complex<br>problems and<br>design step-by-<br>step solutions) | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

(V4, A) Automated Optimisation methods and how humans feel about them

The aim of this questionnaire is to see how much humans trust automated optimisation methods, and whether they can interface with them effectively.

If you have any questions, or would like to revoke your answers, please contact [g.a.catalano@stir.ac.uk](mailto:g.a.catalano@stir.ac.uk).

**This questionnaire is expected to take you around 30 minutes, and you'll be rewarded a £10 Amazon voucher upon completion.**

Please note that to be eligible for this questionnaire, you have to be either:

- a student at the University of Stirling, with some numeracy skills
  - an employee of British Telecom
- (and you have to be over 18)

(The consent form for this study can be found here: <https://www.maths.stir.ac.uk/~gac/ConsentForm.pdf>)

General Information and consent

1. Are you eligible to participate in this study? \*

To participate in this study, you should fit all of the following:

- student at Stirling, or an employee at BT
- You've not filled out this questionnaire before

- ☐ Yes  
☐ No

2. In order to be rewarded the £10 voucher, you'll have to complete all the sections of the questionnaire, and you'll receive the reward by email.

(Note that they are sent manually and will not be sent immediately)

Do you consent to the terms of this study, and to participate in the experiment? \*

The information you'll provide in this questionnaire is:

- email (to send the voucher)
- your answers to the experimental task

- ☐ Yes  
☐ No

3. Please enter your email (only relevant for University of Stirling Students) \*

**(IMPORTANT)** It will only be used for sending the voucher, and won't be shared with others.  
**(IMPORTANT)** It has to be your student email

Please enter text that contains stir.ac.uk

10. How familiar are you with Automated Optimisation Methods?  
These are the methods used to solve optimisation tasks, and they include **Genetic Algorithms**, **Evolutionary Algorithms**, **Hill Climbing** etc...  
For this questionnaire we don't assume that you know what they are or how they work. All you need to know is that they are used to solve optimisation tasks, but are not always guaranteed to give the best or correct answer.

Have you heard of these methods before? \*

- ☐ Not familiar at all
- ☐ I have heard of them, but I don't know how they work
- ☐ I have studied / I am currently learning about them
- ☐ I have used them in the past
- ☐ I am well versed in them

11. Imagine a situation where you are presented with an optimisation task (for example "find the X which gives the best Y"), and your options are:

- **solve it by hand** using your understanding, and trial and error
- **letting a computer solve it** by using an automated optimisation method

How much do you agree with the following statements? \*

|                                                                | Strongly disagree     | Disagree              | Neutral               | Agree                 | Strongly agree        |
|----------------------------------------------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| I prefer to solve tasks by myself rather than using a computer | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I trust that automated methods can produce the results I need  | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I double check the results produced by automated methods       | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

12. How much experience do you have with **Job Scheduling** (for example in assigning working days and rotas to a set of workers)? \*

- ☐ None
- ☐ Very little
- ☐ Some
- ☐ Plenty

## How you feel about technology and automated optimisation methods

Here we'll assess how confident you feel about certain technologies.

8. How do you feel about technology in general? \*

|                                                                                 | Strongly disagree     | Disagree              | Neutral               | Agree                 | Strongly agree        |
|---------------------------------------------------------------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| I am <b>confident</b> in using most technology                                  | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I trust technology to produce <b>useful</b> results                             | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I think that the usefulness of technology is <b>overrated</b>                   | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I would trust technology to <b>decide my work rota</b>                          | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I would trust technology to decide <b>whether I'm suitable for a job</b> or not | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I would trust technology to <b>diagnose me</b> with serious diseases correctly  | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

9. How familiar are you with **Optimisation Tasks**?

These are situations where the aim is to "**find the settings which yield the best score**".  
Some example optimisation tasks could be:

- Finding the most robust timetable for modules
- Finding the least expensive travel plan
- Finding the most efficient wind turbine shape

Are you familiar with optimisation tasks? \*

- ☐ Not familiar at all
- ☐ I understand what they are but I've not studied them before
- ☐ I have studied / I am currently learning about them
- ☐ I have worked them in the past
- ☐ I am well versed in them
- ☐ Other

16. How **confident** did you feel in providing an explanation to the affected workers? \*

- ☐ Extremely confident
- ☐ Somewhat confident
- ☐ Neutral
- ☐ Somewhat not confident
- ☐ Extremely not confident

17. Please select how much you agree with the following statements \*

|                                                    | Strongly Disagree     | Disagree              | Neutral               | Agree                 | Strongly Agree        |
|----------------------------------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| I am <b>confident</b> in the ordering I chose      | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I <b>understand</b> why the changes are disruptive | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I found it <b>easy</b> to choose the ordering      | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

18. Please select how much you agree with the following statements regarding the explanation sheet on **how the optimal arrangement was found** \*

|                                                         | Strongly Disagree     | Disagree              | Neutral               | Agree                 | Strongly Agree        |
|---------------------------------------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| The explanations were <b>helpful</b>                    | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| The explanations were <b>easy to understand</b>         | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I believe the explanations were <b>accurate</b>         | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| The explanations made me <b>trust</b> the solution mode | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

19. Is there any more feedback you'd like to give about the additional explanations?

## First Task

The link below contains a description of the situation you'll be working with.  
(link to problem definition sheet) <https://www.maths.str.ac.uk/~gac/V4Instructions.pdf>

Additionally, you are given some material to help you understand how the computer found that optimal solution:  
(link to explanatory material) [https://www.maths.str.ac.uk/~gac/V4Explanation\\_tgdm.pdf](https://www.maths.str.ac.uk/~gac/V4Explanation_tgdm.pdf)  
(The additional explanations are meant to make the task easier and we suggest that you make use of them, but please don't share them!)

13. Considering the problem description and the proposed rota assignments, please answer the following \*

|                                                                               | Strongly disagree     | Disagree              | Neutral               | Agree                 | Strongly agree        |
|-------------------------------------------------------------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| I <b>understand</b> the general structure of the problem                      | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I understand the <b>additional explanation material</b>                       | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I <b>trust</b> that the solution produced by the computer is good             | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I believe that I <b>could make</b> a better rota assignment than the computer | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

14. Due to unforeseen circumstances, a change needs to be the arrangement found by the computer. You have 4 possible choices available, and you have to determine which ones will cause the disruption (meaning that the difference between weeks will increase).

Please order the following changes, from the **least disruptive** to the most **disruptive**. In other words, the change which you think will cause the **smallest increase in difference should be at the top**. \*

Put Adam on A

Put Isaac on B

Put Adam on A and Isaac on B

Put Isaac on B and Olivia on A

15. Imagine that you have to explain your answer to the workers: what would you say? \*

Second Task

You'll be working with the same problem, but this time the workers and their skills are different (and therefore the computer found a different optimal arrangement).  
The task is described here: <https://www.maths.sir.ac.uk/~gac/V4Binstructions.pdf>

(This time you are not provided with the material that explains how the optimal solution was obtained)

21. There are 5 workers that would like to change their rota, but you can only satisfy the wishes of 2 of them. Your task is to decide which 2 workers get their desired rota, with the aim of not introducing too many differences between the weeks.

Please select the 2 workers that will get their wishes fulfilled:  
(keep in mind that your 2 choices might have a compound effect together)

Please select 2 options.

- ☐ Alice to D
- ☐ Brandon to D
- ☐ Kevin to C
- ☐ Lucas to D
- ☐ Phoebe to C

22. Suppose that you have to explain to the 3 workers that DIDN'T get assigned their preferred rota, how would you explain to them your decision? \*

23. Please select how much you agree with the following statements \*

|                                                         | Strongly Disagree     | Disagree              | Neutral               | Agree                 | Strongly Agree        |
|---------------------------------------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| I am <b>confident</b> in the workers that I chose       | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| I have <b>good reasons</b> to have chosen those workers | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

20. You can give more feedback about this task if you'd like

# G

## APPENDIX G: ETHICS APPROVAL

---

This research project was approved by the University of Stirling General University Ethics Panel. Proof of ethics approval is shown in Figure G.1.

Dear Giancarlo Catalano

Ethics Application Form : ENDS: Explainability of Non-Deterministic Solvers

Ethics Reference Number: EC\_2024\_18852\_13620

Thank you for your submission of the above ethics application.

The ethical approaches of this project have been approved and you can now proceed with your project.

Figure G.1: Proof of ethics approval