Explainability of non-deterministic solvers: explanatory feature generation from the data mining of the search trajectories of population-based metaheuristics.

FYVIE, M.

2024

Explainability of Non-Deterministic Solvers: Explanatory Feature Generation from the Data Mining of the Search Trajectories of Population-Based Metaheuristics

Martin Fyvie

School of Computing
Robert Gordon University

A thesis submitted in partial fulfilment of the requirements of Robert Gordon University for the degree of
Doctor of Philosophy

May 2024

Declaration

I hereby declare that this thesis is a record of work undertaken by myself. That it has not been the subject of any previous application for a degree and that all sources of information have been duly acknowledged.

Martin Fyvie
May 2024

Acknowledgements

I would like to extend my sincerest thanks to my principal supervisor, Professor John A.W McCall. Our long and often meandering conversations have guided this work in more ways that I can appreciate, and the wealth of knowledge and advice given has been invaluable. In addition, I would also like to thank my supervisory team at RGU - Dr Lee A. Christie and Dr Alexandru-Ciprian Zăvoianu - without their guidance and support this work would not have been possible. I would also like to give my thanks Dr Alexander E.I. Brownlee at Stirling University and Dr Russell Ainslie at the BT Group for their direction and kind words of support which were always appreciated.

I wish to thank the National Subsea Centre, the Robert Gordon University, the BT Group and the Data Lab for their training, funding and support throughout the PhD journey. These invaluable resources have allowed me to travel and experience academic life in a way that would not be possible without their support.

Lastly, I would like to express my love and gratitude to my family. These last four years have had many ups and downs but through it all, you have always been there to support and encourage me.

Abstract

Evolutionary algorithms (EAs) are the principal focus of research study in Evolutionary Computing (EC). In EC, naturally occurring processes designed to drive success in nature are simulated for a similar purpose in numerical optimisation. Such processes include natural selection, genetic mutation and breeding of parents to pass on beneficial traits. EAs are a family of algorithms that utilise this diverse set of processes to navigate an optimisation problem's landscape using generations of solutions that evolve. This diverse set of processes often leads EAs to be considered "Black-Box" in that, due to the stochastic nature of their internal operators, generating an understanding of the reasoning behind EA decisions can be difficult.

In optimisation and artificial intelligence (AI), the field of explainable AI (XAI) has grown significantly as machine learning, systems that mimic human reasoning and other AI systems have continued to be adopted into more and more user-critical applications. XAI as a research area aims, among many things, to aid in gaining a better understanding of these decision-making processes. EAs are often employed as mechanisms within explanation generation techniques such as counterfactual and fuzzy-rule refinement and were, until recently, rarely the focus of study for XAI. While there is no single mathematical theory that extends to all EAs, one commonality that extends to all population-based EAs is their generation of successive populations of solutions. These generations of solutions represent the EA's understanding of the optimisation problem at each specific point in the search and collectively represent the search trajectory of the algorithm.

This thesis aims to expand on current research in the field of XAI by introducing a set of novel XAI methodologies designed specifically for use in deriving explanations directly from the search trajectories of EAs. These explanations take the form of geometrically sensitive features detected by the decomposition of the search trajectories using PCA.

We show that the PCA decomposition of the search trajectories of EAs retains sufficient structure that geometrically derived features, in this case, vector similarities and component loadings, can identify low-level variable interactions and capture a similar level of information in terms of population diversity as the known method of Kullback-Liebler entropic divergence. With this ability to generate the variance-based subspaces and show that sufficient structure is retained, we are able to extend this approach to additional problem representations – real and nominal and expand upon the initial techniques. We propose a set of novel XAI trajectory mining techniques designed to identify variable importance and to highlight differing algorithm behaviour on the same problems by highlighting both variables and components that contribute more than others to solution quality improvements at different stages of the search.

Finally, we also reflect on the research project, identify areas of possible improvement and set out a range of future work to further expand on that results found in this thesis.

Publications

Parts of the work presented in this thesis have appeared in the following publications:

Prizes: Best Application Paper 2023 Winner

- Martin Fyvie, John A. W. McCall, Lee A. Christie, Alexandru-Ciprian Zăvoianu, Alexander E. I. Brownlee, and Russell Ainslie. Explaining a Staff Rostering Problem by Mining Trajectory Variance Structures. In *Artificial Intelligence XL: 43rd SGAI International Conference on Artificial Intelligence, AI 2023, Cambridge, UK, December 12–14, 2023, Proceedings*, page 275–290, Berlin, Heidelberg, 2023. Springer-Verlag. doi: https://doi.org/10.1007/978-3-031-47994-6_27

Journal Papers:

- Martin Fyvie, John A. W. McCall, Lee A. Christie, Alexander E. I. Brownlee, and Manjinder Singh. Towards explainable metaheuristics: Feature extraction from trajectory mining. In *Expert Systems*, page e13494, 2023. doi: https://doi.org/10.1111/exsy.13494
- Martin Fyvie, John A.W. McCall, and Lee A. Christie. "Towards Explainable Metaheuristics: Feature Mining of Search Trajectories through principal component projection". In *SUBMITTED TELO*, pages 89–102. Springer, 2023
- Martin Fyvie, John A. W. McCall, Lee A. Christie, Alexandru-Ciprian Zăvoianu, Alexander E. I. Brownlee, and Russell Ainslie. "Expansion on: Explaining a Staff Rostering Problem by Mining Trajectory Variance Structures". In *SUBMITTED Künstliche Intelligenz (KI) Placeholder*, page x. Springer, 2024

Conference Papers:

- Martin Fyvie, John A.W. McCall, and Lee A. Christie. Non-deterministic solvers and Explainable AI through Trajectory Mining. pages 75–78. CEUR Workshop Proceedings, 2021

- Martin Fyvie, John A. W. McCall, and Lee A. Christie. Towards Explainable Metaheuristics: PCA for Trajectory Mining in Evolutionary Algorithms. In Max Bramer and Richard Ellis, editors, *Artificial Intelligence XXXVIII*, pages 89–102, Cham, 2021. Springer International Publishing

- 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*, GECCO '23 Companion, page 1648–1656, New York, NY, USA, 2023. Association for Computing Machinery. doi: https://doi.org/10.1145/3583133.3596353

Acronyms:

List of Acronyms

ACO Ant Colony Optimisation.

AI Artificial Intelligence.

ANOVA Analysis of Variance.

BBOB Black-Box Optimisation Benchmark Problem Set.

BOA Bayesian Optimization Algorithm.

BT The BT Group.

CART Classification and Regression Trees.

CMA-ES Covariance Matrix Adaptation Evolutionary Strategy.

CoDE Composite Differential Evolution.

CR Crossover Rate.

CV Constraint Violation.

DARPA Defense Advanced Research Projects Agency (United States).

DE Differential Evolution.

DEUM Distribution Estimation Using Markov Random Fields Algorithm.

EA Evolutionary Algorithms.

EC Evolutionary Computing.

EC-XAI Evolutionary Computing for Explainable Artificial Intelligence.

EDA Estimation of Distribution Algorithms.

ELA Exploratory Landscape Analysis.

EP Evolutionary Programming.

ES Evolutionary Strategy.

EU European Union.

FDR False Discovery Rate.

GA Genetic Algorithm.

GDPR General Data Protection Regulation (United Kingdom).

GECCO Genetic and Evolutionary Computation Conference for Optimisation.

GP Genetic Programming.

hBoa hierarchical Bayesian Optimization Algorithm.

ICE Individual Conditional Expectation Plots.

ICGA International Conference on Genetic Algorithms.

ICO Information Commissioners Office.

IoT Internet of Things.

KLd Kullbach-Liebler Entropic Divergence.

LDA Linear Discriminant Analysis.

LIME Local Interpretable Model-Agnostic Explanations.

LON Local Optima Networks.

MCA Multiple Correspondence Analysis.

MIMIC Mutual-Information-Maximizing Input Clustering.

ML Machine Learning.

MOEA Multi-Objective Evolutionary Algorithm.

MOEAD Multi-Objective Evolutionary Algorithm based on Decomposition.

MSC Mean Squared Cosine.

NDE Neighbourhood-Based Differential Evolution.

NMF Non-Negative Matrix Factorization.

NSGA-II Non-Dominated Sorting Genetic Algorithm II.

PBIL Population-Based Incremental Learning Algorithm.

PC Principal Component.

PCA Principal Component Analysis.

PDP Partial Dependency Plots.

PLOS-Net Network of Pareto local Optimal Solutions.

PPSN International Conference on Parallel Problem Solving.

PS Pareto-Optimal set of solutions.

PSO Particle Swarm Optimisation.

PYMOO Python Multi-Objective Optimisation.

SA Simulated Annealing.

SHADE Success-history based parameter adaptation for differential evolution.

SHAP Shapley Additive Explanations.

SPEA2 Strength Pareto Evolutionary Algorithm 2.

STN Search Trajectory Networks.

SVD Single Value Decomposition.

t-SNE t-Distributed Stochastic Neighbour Embedding.

TCDT Transformed Complete Disjunctive Table.

UMAP Uniform Manifold Approximation and Projection.

WRBO Weighted Rank Biased Overlap.

XAI Explainable Artificial Intelligence.

Glossary:

The following list contains the definition of commonly used terms in Chapters 3 onwards of this thesis.

x	Candidate solution
x_n	n-th variable of solution x
N	The problem size
X	A generation of solutions (x)
g	The number of generations in an optimisation run
m	The number of components used to define a PCA-derived subspace
T	Search Trajectory - All visited solutions in an optimisation run
f	A fitness function.
$f(x)$	The fitness of solution x
f_g^*	Fitness of the best-found solution at generation g
Δf^*	Delta in fitness between the best-found in the starting generation and best-found overall in a run
U	Left singular values from SVD
Σ	diagonal matrix of singular values from SVD
P^T	Transpose of P containing right singular values from SVD
P	Principal components (Column values of P^T)
$\widetilde{T}$	The projection of Trajectory (T)
$\widetilde{X}$	The projection of generation (X)
$\widetilde{x}$	The projection of solution (x)
C_g	Geometric centre of all solutions in generation g across each axis
$C_i^{'}$	Vector of mean variable contributions across a given component (p_i)
V_j^i	Binary vector of variable j's projected score signs across component i
$G^i(x_j)$	Binary vector of variable j's alignment across component i
PG^i	Vector of proportion of times G^i is true across component P_i in a generation of solutions.
$PC_{x_{nc}}$	Principal coordinate of x_n in category c in MCA
PC_{x_n}	Principal coordinate of x_n across all categories MCA
λ_m	Eigenvalue of component p^m in MCA
wp	Ranked Biased Overlap weighting parameter
Pr, Q	Vector of marginal probabilities of two populations

List of Figures

List of Tables

Table of Contents

Chapter 1

Introduction

1.1 Overview

Evolutionary Computing (EC) is a well-established collection of computational techniques inspired by the process of natural evolution. Modern EC, according to Bäck. et al [8] can broadly be considered the merging of three distinct, convergent evolutions of computational optimisation strategies - Evolutionary Programming (EP) [9, 10], Genetic Algorithms (GAs) [11] and Evolution Strategies (ES) [12, 13]. Major collaboration within these areas in the 1980s and 1990s can be seen in the creation of the International Conference on Genetic Algorithms (ICGA) in 1985 and the International Conference on Parallel Problem Solving (PPSN) in 1990. This has since led to the creation of additional sub-disciplines of EC including Evolutionary Algorithms (EAs) such as Particle Swarm Optimisation (PSO) [14], Differential Evolution (DE) [15], Estimation of Distribution Algorithms (EDAs) [16], Ant Colony Optimisation (ACO) [17], Genetic Programming (GP) [18] and a plethora of related algorithms using other nature-inspired metaphors.

Evolutionary Algorithms can utilize population-based, stochastic search methods to navigate optimisation problem landscapes. This is often achieved through the application of operators which typically mimic processes such as natural selection, genetic exchange and gene mutations with the aim of, over time, iteratively improving a set of candidate solutions. This process progresses (ideally) towards a global or set of globally optimal solutions to said problem. Not all EAs employ the same set of operators - EDAs for example, replace the crossover operator typically used in GAs with the sampling of the solution pool to create a probabilistic model. This model is updated and re-sampled to create successive generations of solutions. PSOs do not typically involve a selection, crossover or mutation process to mate and alter solutions - instead, they track the position, velocity and best-found solutions in the problem landscape to update each generation.

While there are many differences between these EAs in their implementation, a trait shared across many is their use of "high-level search metaphors, often nature-inspired, implemented as complex, non-deterministic sequences of operations on solution populations, leaving the solution process intractable to explanations based on analogies to human reasoning." [2]. The problem of intractability is common across many EAs and other AI systems, leading to the use of terms such as "Black Box" and "White Box" optimisers.

Each of these terms refers to the level of exposure an end-user is given to the inner workings and their inherent level of understandability. As noted in [19] A Black-Box solver provides little to no exposure of their implementations and is often considered difficult to explain even by experts [20, 21] whereas White-Box solvers aim to expose as much as possible, tending towards more interpretable and understandable processes [22].

In recent years, population-based metaheuristics such as EAs have seen an increase in applications in areas such as Transport and Logistics [23, 24], Medical applications [25, 26] and Engineering [27, 28]. This growth in safety-critical industries means that trust in these systems is increasingly crucial, and designing such systems according to a design philosophy with interpretable results and operations as a focus is highly recommended. This ensures that stakeholders (or decision-makers) can understand and trust the AI's outputs and decision-making processes. These developments show a growing need to combat the issue in non-interpretable systems that deal with sensitive data, such as Black-Box systems whose decision-making processes require expert-level knowledge to understand.

At the same time, the field of Explainable Artificial Intelligence (XAI) has also seen a considerable increase in attention. XAI aims to increase understanding of these complex processes and aid in building trust between the decision-makers and the systems. The growth in AI system usage can also be seen in the greater level of legislative scrutiny of their level of interpretability, especially when dealing with public data. In the UK, General Data Protection Regulation (GDPR) Article 13.2.f and Recital 71, which concern the right to be informed about data usage, may be extended to cover interactions with AI systems like those discussed in this thesis. These articles require providing "meaningful information about the logic involved, as well as the significance and the envisaged consequences" [29, 30]. This consideration has recently become a core principle in AI, as evidenced by its inclusion in the 2023 European Commission publications on Trustworthy AI [31]. The Act outlines detailed requirements for transparency and explainability of AI systems, aligning with the principles of XAI which seeks to make AI decisions understandable to humans. However, it is noted that there could be significant differences between the solutions offered by XAI and the requirements stipulated in the EU AI Act, such as the lack of an explicit definition of transparency [32]. The lack of a clear definition of common XAI terms such as transparency and its applicability to different AI implementations is not unique to the EU act, as will be discussed in this thesis. The establishment of these guidelines has influenced the AI community towards renewed development and implementation of existing and novel XAI techniques.

Within the XAI community, EAs are an often-used tool to aid in developing and refining systems designed to increase the understandability of AI systems [33]. They were until recently, however, rarely the focus of this research. This is in part due to the stochastic, population-based nature of their inner workings mitigating against most standard XAI techniques. A central problem in EC and a significant motivator for the research in this thesis is that, due to their design, the same EA may give different search results to the same optimisation problem. This issue is found throughout the field of XAI as a whole - the need to understand both the how and why of solution selection by EAs must account for the algorithm's search behaviour as well as solution quality in terms of fitness.

Evolutionary Computing for eXplainable Artificial Intelligence (EC-XAI), a subset of XAI research, focuses on such systems to create novel solutions to this problem and to help extract meaning and understanding from such EAs. As population-based EAs perform their search, the process generates a path or trajectory, representing the locations within the problem landscape that the algorithm and its populations have explored. This search trajectory reflects the EAs implicit current model of the fitness function. This thesis will focus on the exploration of these EA search trajectories and their potential to provide interpretable explanations regarding solution quality drivers and algorithm search behaviour differences.

1.2 Aim of The Thesis

The purpose of this thesis is to investigate whether it is possible to perform data mining on the set of solutions visited by an evolutionary algorithm for explanatory feature creation. We aim to discover whether the search trajectories of population-based evolutionary algorithms contain sufficient information and latent structure that can be extracted and used to help improve the understanding of the search process and solution quality drivers. To this end, the following research questions were formulated to guide this work:

Research Questions

(Q1) In what ways can and do EAs benefit from XAI?

(Q2) How can we formalize a method of solution-population structuring such that novel XAI features can be extracted from the algorithm search path?

(Q3) Can the search trajectories of EA, across a known fitness gradient, be mined for novel explanatory features that relate some level of knowledge regarding algorithm search behaviour to an end-user?

(Q4) Can a method of explanatory feature mining from the search trajectories be formalised and be made applicable to a range of optimisation problem representations?

Research Objectives

To answer the research questions, the following objectives were set:

(O1) To identify a subset of widely used evolutionary algorithms with a representative variety of search mechanisms that would benefit from the application of XAI techniques. This objective aims to address the research question (Q1). This research aims to identify a subset of population-based EAs that are commonly used in both industry and academic studies that would represent a significant portion of the search strategies utilized by both. Work towards this objective can be found in Chapter 2.

(O2) To develop a definition that formalised the structure of EA search that can be used to structure future experimentation and analysis. Chapter 3 introduces this definition which would allow for a direct mapping of an EA search path to the features derived from such paths and is a move towards answering the research question (Q2).

(O3) Perform exploratory analysis and data mining on the search trajectories created by EAs to answer question (Q3). This aims to identify possible features and artefacts contained within these trajectories that can help shed light on the search process and aid in end-user understanding of the main drivers of solution quality to the algorithms used. Chapter 5 introduces this work as well as serves as a basis for later developments.

(O4) Based on the results of the exploratory analysis and data mining, identify a subset of possible end-users that would be most suitable to target the generation of explanations. This objective is designed to also aid in answering question (Q3) The identified end-users are detailed in Chapter 2 in which they were selected from a range of possible user types.

(O5) Towards answering question (Q4), this objective is to develop a set of methods capable of leveraging any features identified and transform the knowledge gained into a level of explanation suitable for the targeted end-users. The work in Chapters 5, 6 and 7 spans the development of such methods and their application to a range of problems while considering the users identified in objective (O4).

1.3 Contributions

The following outlines the original research and contributions to the XAI and EA bodies of knowledge generated throughout this project.

Contribution 1

A literature review of the current XAI landscape that identifies the growing branch of EC related XAI methodologies. This has helped identify the "evolutionary gap" – the gap in knowledge relating to the development and application of EC-specific XAI methods into which the work in this thesis sits. This contribution aids in answering research question (Q1) by achieving objective (O1). The review of current literature and XAI methods, presented in Chapter 2, highlights a representative set of EC algorithms, introduces current XAI methods and discusses their applicability to EC algorithms. This contribution also addresses objective (O4) in which a set of common end-user types are discussed, from which the "User" and "Developer" are selected as the focus of explanation generation. Sections of this work are planned for publication as part of an EC-XAI review paper submitted to a reputable journal.

Contribution 2

By structuring the search trajectory in the form introduced in Chapter 3, this thesis shows that these novel, PCA-derived subspace angular metrics can be developed. These metrics can detect a comparable level of population diversity change as KLD with the added benefit of extracting low-level variable interactions from the weightings of the resulting hyperplanes used to define the subspace for binary problems. This contribution addresses research questions (Q2) and (Q3) by achieving objectives (O2) and (O3) through an initial investigation into the concepts and application to a set of binary benchmark problems. The metrics developed towards this contribution are shown in Chapter 4 and the results supporting the contribution are found in Chapter 5. This contribution is further supported by publications in the BCS SGAI International Conference on Artificial Intelligence and in the journal Expert Systems.

Contribution 3

This contribution relates to the adaptation of trajectory mining techniques to continuous space problems and introducing metrics for variable importance and alignment detection. By modifying the analysis approach, this work measures the impact of individual variables at various stages of the search process, identifying variables that significantly influence solution quality. This allows tracking of how variable importance shifts over time and enables comparisons across algorithms. The development of the variable alignment metric detects changes in variable importance across specific PCA-derived subspace axes, highlighting algorithmic behaviour differences on identical problems. These metrics provide detailed insights for "Developer" users, offering high-level explanations of algorithm behaviours and their discrepancies. This research, which addresses research question (Q4), is detailed in Chapter 6 and submitted for publication as part of a Evolutionary Computing and Explainable AI survey paper by a number of authors including myself to Transaction on Evolutionary Learning.

Contribution 4

Demonstration of the applicability of search trajectory mining to nominal spaces and to real-world problem applications. This contribution involves the use of Multiple Correspondence Analysis (MCA) to allow the application of our methods to a nominal-encoded version of a BT-specific staff roster allocation problem. This work identifies both high and low impact sets of variables, in this case sets of working hours, in order to be presented to the “User” in an interpretable manner. To achieve this, a ranking system is introduced that highlights these variable sets. By doing so, the contributions of this work include identifying what workers and rosters can be altered to make minimal impact changes to solutions to meet additional, unexpected requirements. This contribution completes the work towards answering research question (Q4) and achieves objective (O5). Research towards this contribution is shown in Chapter 7 and is supported by publications in the BCS SGAI International Conference on Artificial Intelligence, the Genetic and Evolutionary Computing for Optimisation XAI workshop and planned publication in jounral Künstliche Intelligenz.

Use Cases

The techniques used in Chapter 5 would allow the Developer to compare changes in population diversity using two metrics that utilise different fundamental principles. This provides the ability to detect differing search behaviours between EAs using these metrics. These techniques also allow the Developer to observe whether a given algorithm can detect low-order variable interactions key to the structure of the optimisation problem when compared to others. This can aid in debugging and algorithm design by highlighting any issues in structure detection.

The work in Chapter 6 provides the Developer the ability to compare algorithm search behaviours in terms of variable importances at specific stages of the search. These techniques allow for the direct comparison between EAs in terms of what variables are considered key to improving solution quality as extracted from their trajectories post-hoc. This provides the ability to attribute an EAs search performance to its ability to detect and exploit key variables when compared other EAs.

The contributions shown in Chapter 7 relate to both the Developer and User, in which we adapt our techniques to be used on an instance of a real-world problem. The techniques showcased in Chapter 7 allow the Developer to gain further insight into what the EA considers key drivers towards solution quality. By using a ranking system, the User is presented with a more interpretable set of results highlighting low and high impact variables. This supports the User in terms of decision making with information regarding what variables should be maintained and which can be altered with minimal impact to overall solution quality while accommodating factors not captured in the fitness function. These could include covering absences and introducing further flexibility to staff allocation.

1.4 Thesis Outline

The rest of the thesis is structured as follows:

- Chapter 2 provides an overview of metaheuristic search, evolutionary computing and explainable artificial intelligence. This section introduces the "Evolutionary Gap", a gap in the newly forming area of Evolutionary Computing for XAI (EC-XAI) knowledge base that supports the need for the development and application of XAI techniques directly to the output of EAs.

- Chapter 3 presents an introduction and detailed explanation of how we define our search spaces and problem encoding and presents a notation that will be used throughout the remainder of the thesis.

- Chapter 4 gives an overview of the methodologies used to explore and mine the search trajectories of the selected evolutionary algorithms. This chapter introduces how we structure a search trajectory, the main methods used to decompose and mine these trajectories and the techniques used to generate and compare the experimental results.

- Chapter 5 compares the results of two analysis methods applied to search trajectories generated by two evolutionary algorithms (EAs) on a set of binary benchmark problems. The first method utilizes entropy-based analysis, while the second leverages spatial-based angular metrics derived from Principal Component Analysis (PCA). This chapter highlights the overlap between the two when measuring information gain as well as the PCA-based technique's ability to detect low-order problem structure.

- Chapter 6 presents the results of the data mining of the search trajectories generated by a selection of metaheuristics. These trajectories are created via the optimisation of the well-known set of real-valued "Black-box Optimization Benchmarking" problems. In this section, we explore the application of trajectory mining techniques using PCA decomposition to measure Variable Alignment - a measure of a variable's influence in determining the position and direction of search in PCA-derived subspaces. Also explored are the contributions of each variable at different stages in the search.

- Chapter 7 presents the results of the application of the analysis techniques developed in this thesis to a real-world problem - the Minimally Disruptive Roster Allocation problem developed with BT to approximate a real-world system. The problem is in the discrete space representing a real-world problem supplied by BT covering worker roster allocations to minimise the disruption caused to current staff. This chapter covers the use of Multiple Correspondence Analysis to adapt our techniques to work with discrete problems. In this section, we highlight the results by comparing detected variable importances between our methods and another variance-based method ANOVA. The explanatory features generated are presented in a form developed with the identified end-users requirements in mind.

- Chapter 8 discusses the thesis' progress towards answering the outlined research questions, how the objectives have been achieved and reflects on any limitations to the research project. This chapter also provides an overall summation of the findings and accomplishments of the project and introduces some potential avenues of future work to extend our findings.

Chapter 2

Literature Review

2.1 Introduction

In the context of our research, two main areas need to be discussed, those being optimisation through metaheuristic search and explainable artificial intelligence. This chapter provides a comprehensive introduction to these areas, laying the groundwork for the research presented in subsequent chapters. The first part of this chapter focuses on optimisation and metaheuristic search. It begins with the low-level heuristics of local search and hill climbing. This is then expanded upon by introducing population-based metaheuristics and some of the most commonly used types. This includes Genetic Algorithms (GAs), Estimation of Distribution Algorithms (EDAs) and Differential Evolution (DE). The second part of this chapter reviews the literature and research surrounding XAI and how this area of research has been developing in recent years. This section covers initial work done in XAI, how this has progressed to cover many types of Machine Learning (ML) and AI and introduces a subset of this work - EC for XAI - an area of research focused on the development and application of new techniques to Evolutionary Computing (EC) and EAs while also outlining the common taxonomies used to describe and place XAI research. Finally, this section discusses the gap in the literature specifically surrounding the development and application of XAI techniques to population-based metaheuristics.

2.2 Metaheuristic Methods for Optimisation Algorithms

In optimisation there are many methods in which a complex problem can be solved, both analytically or heuristically. Of these, there are two concepts important to this thesis - Heuristic and Metaheuristic optimisation strategies. As noted by Sorensen "...A metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms." [34]. The defining aspect mentioned here is the problem-independent nature of a metaheuristic. Heuristics can be considered a search strategy that exploits the problem-specific information regarding structure and so is not generalisable to a range of problems. An example of this would be a greedy search based algorithm such as ID3 for decision tree branch and node generation [35]. This process searches for "good enough" solutions, potentially being trapped by locally better solutions as it decides which attribute to split the data by.

Contrasting this, a GA is a population-based metaheuristic capable of being adapted to work on a range of problem representations. The GA's ability to adapt to the landscape of the optimisation problem in order to avoid locally optimal solutions, much like many evolutionary algorithms, takes the form of simulating naturally occurring evolutionary processes. It is this ability to be adapted to a range of problems and overcome issues such as locally optimal solutions that draws our research towards the exploration of the search behaviours of these algorithms, both evolutionary-based and otherwise.

2.2.1 Metaheuristic Optimisation

In metaheuristic optimisation, the objective function $f(x)$ is often a mathematical function that represents the relationship between variables, constraints and parameters of the optimisation problem and the values in the objective space. This defines the goal (either maximisation or minimisation) of finding the set of variable values that optimise this objective. To solve such a function, the solution generated by an algorithm requires a representation that can be manipulated by its internal mechanisms. String encoding is a common method used to represent solutions in a structured and manageable format for such problems. As an example, an optimisation problem solution x consisting of a set of discreet variables $x_1, x_2, \ldots, x_n$, can be assigned a fitness function value $f(x)$ based on its specific variable values.

It is possible for an optimisation problem to have two or more objectives. Usually, when there are two or three objectives that aim to be optimised it is referred to as Multi-Objective. With a higher number of objectives, the term Many-Objective is adopted. In these cases, the optimisation algorithm aims to assign values such that the combination of $[f1(x), f2(x), \ldots fn(x)]$ generates solutions to discover the optimal trade-offs between individual objective functions. As there are multiple, usually conflicting, objectives it can be stated that there is no guarantee that a single solution to the problem will satisfy all requirements. In these cases, the target is to discover a Pareto-Optimal set of solutions that represent the best possible trade-offs between the varying objectives. The Pareto-Optimal set (PS) is an aggregation of all candidate solutions, such that they all have the property of not being fully dominated [36] - there is no solution in the decision space that is better, with regard to all the considered objectives, than those in the PS [37]. This provides decision-makers with a range of solutions satisfying multiple scenarios depending on objective importance to the decision-maker.

Problems can be broadly categorized into different spaces based on the nature of the variables involved, primarily discrete ($\mathbb{Z}$) and continuous ($\mathbb{R}$) spaces. In discreet spaces, the set of all possible solutions is usually finite and each variable value is distinct such as an integer or binary value. In continuous spaces, the set of all solutions is considered infinite however in practice, this is usually reduced to a (often prohibitively large) discreet space due to the limitations of digital representations of real-values [38]. This can also be seen should the solution representation ultimately be implemented into a physical system - there are limitations to the level of precision with which manufacturing processes can be conducted cost-effectively.

2.2.2 Problem Landscapes

In optimisation each solution, both valid or not, represents a position in the search space. The set of all possible feasible solutions that satisfy all problem constraints is known as the domain. Within the search space, the objective function codomain, which evaluates the fitness of a solution, can be visualised as a landscape. This visual representation, commonly referred to as a search landscape, depicts the space that an optimisation algorithm explores [39].

An illustration of these landscapes, characterized by peaks, valleys, and plateaus, can be seen in Figure 2.1 which shows the landscape of a two and three-dimensional codomain of the Rastrigin function which is detailed further in Chapter 3.5.

Optima and Fitness Gradients

Given a search space S, for any point $x^* \in S$, the neighbourhood of x is defined as a set of points in S that are in close proximity to x according to some metric. The neighbourhood ($N(x)$) yields a subset of S representing the points adjacent to x. The definition of neighbouring solutions can vary depending on the problem and the structure of the search space. For instance, in a Euclidean space, the neighbourhood of x could be defined as all points within a certain radius r of x, as defined by the distance d(x,y), $N(x) = \{y \in S | d(x,y) \leq r\}$. In discrete or combinatorial spaces, the neighbourhood might consist of all solutions obtainable by applying a single elementary operation to x, like swapping, inserting, or deleting an element.

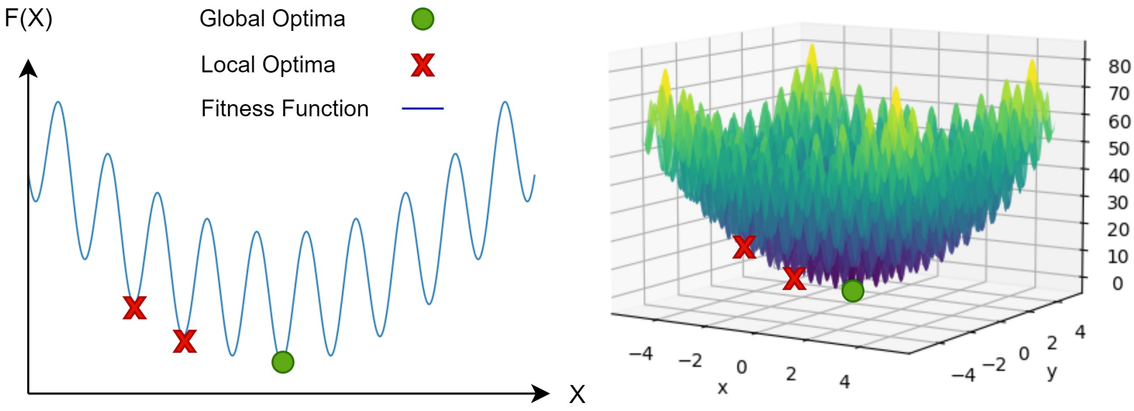


Fig. 2.1 Rastrigin Global and Local Optima Example.

A local optimum is a point $x^* \in S$ where the objective function $f(x^*)$ is greater than or equal to the function values of its neighbours. This means that for all $x \in N(x)$, the fitness of the local optimum is greater than all solutions in its neighbourhood - $f(x^*) \geq f(x)$. However, a local optimum may not be the best solution overall.

A global optimum is a point $x^* \in S$ where the objective function $f(x^*)$ is greater than or equal to all other points in the search space. Mathematically, for all $x \in S$, we have $f(x^*) \geq f(x)$. A global optimum represents either the single best solution to the problem or one of a set of equally optimal solutions if multiple exist.

Shown in Figure 2.1 is the 1-dimensional and 2-dimensional domain of the Rastrigin optimisation problem as mentioned earlier. In the Figure, the global and a selection of locally best solutions are highlighted to demonstrate these concepts. As the dimensionality of the problem increases, so does the complexity of the search landscape, often requiring a more advanced form of heuristic to optimally or near-optimally solve the problem within a given time constraint.

In optimisation problems, the concept of a fitness gradient helps to navigate the search towards optimal solutions by indicating the direction in which the fitness improves. In metaheuristic search the concept of fitness gradient is key to the function of many search strategies. The fitness gradient is used to help these algorithms traverse the problem landscape towards optimal solutions by measuring the direction in which the greatest solution quality improvements can be found across n dimensions in $\mathbb{R}^n$. This is often achieved by calculating how the fitness function changes with regard to movement in a given dimension and moving in the direction of the steepest ascent. Shown in Equation (2.1) is the method for calculating the fitness gradient vector in continuous spaces.

$$Df(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right) \tag{2.1}$$

Here, $Df(\mathbf{x})$ is the fitness gradient at a given point x. The partial derivative is calculated to measure how the fitness function changes with movement across dimension x_n. Depending on whether the problem was a minimisation or maximisation problem, the algorithm would move away from, or in the direction of the gradient respectively. Some adaptation is needed for discreet spaces in which the difference between the fitness of neighbouring solutions must be considered, as shown in Equation (2.2)

$$\begin{aligned} Df_{\mathbf{x}} &= (\Delta f_{x_1}, \Delta f_{x_2}, \ldots, \Delta f_{x_n}) \\ \Delta f_{x_i} &= f(x_1, \ldots, x_i + \Delta x_i, \ldots, x_n) - f(x_1, \ldots, x_i, \ldots, x_n) \end{aligned} \tag{2.2}$$

Here we show how to approximate the fitness gradient over a single dimension i in a discreet space $\mathbb{Z}^n$ is calculated as Δf_{x_i}. Also shown is how the vector $\Delta f_{\mathbf{x}}$, which represents the change in fitness across all n dimensions in the space is calculated.

Constraints

It is often necessary to restrict the possible search area in a given problem to a set of feasible solutions using constraints - a set of applied restrictions dependent on the problem requirements. Shown in Equation (2.3) is an adapted version of an optimisation problem as in [40].

$$\begin{aligned} \text{Minimize: } 5x_1 + 3x_2 + 8x_3 \text{ s.t:} \\ x_1 + x_2 + 2x_3 = 4 \\ x_1, x_2, x_3 \geq 0 \end{aligned} \tag{2.3}$$

Here, the fitness function $f(x)$ is to minimise the sum of $5x_1 + 3x_2 + 8x_3$ subject to a set of constraints. These typically come in two forms - Soft Constraints and Hard Constraints. The constraints shown in the problem are hard constraints that restrict the possible area of feasible solutions. Any solution that violates either of these would result in an infeasible solution that would not be accepted. Figure 2.2 shows how these constraints restrict the search area to a triangular plane in 3 dimensions of all possible non-negative variable values. In this problem, the optimal solution would lie at point Q with a fitness value of 12.

As outlined in a recent survey on constraint handling techniques for metaheuristics [41] there are many different ways in which an EA can deal with constrained problems including those outlined in this chapter. In their work, they show 49 variants of popular and novel methods found in state-of-the-art metaheuristics. Here, we will introduce three of the main

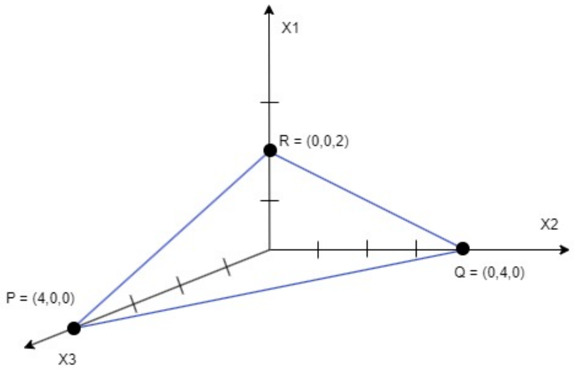


Fig. 2.2 Constrained Search Area from [40]

methods typically found in EAs.

Fitness Function Adjustment - Penalty

One method of dealing with constraints in metaheuristics is an adjustment of the fitness function with a penalty. Soft constraints are a form of restriction that penalises, rather than disregards, an infeasible solution. This is done by a penalty function that coerces the search when it has strayed too far from the initial problem specification. This can be achieved with either an excess or slack property. A slack variable is added to a greater than or equal constraint and an excess variable is subtracted from a less than or equal constraint, converting either into an equality constraint. These variables represent the possible variability a solution can take.

Equation (2.4) shows an example of an updated version of the problem, defined in Equation (2.3), with an excess variable (P) representing a penalty function that penalises any solution in which the sum of the variables is greater than 12.

$$\begin{aligned} \text{Minimize: } 5x_1 + 3x_2 + 8x_3 + \lambda P \text{ Subject to:} \\ x_1 + x_2 + 2x_3 = 4 \\ x_1, x_2, x_3 \geq 0 \\ \text{Where: } P = \begin{cases} 0 & \text{if } x_1 + x_2 + x_3 \leq 12 \\ x_1 + x_2 + x_3 - 12 & \text{if } x_1 + x_2 + x_3 > 12 \end{cases} \\ \text{and } \lambda \text{ is a penalty coefficient.} \end{aligned} \tag{2.4}$$

Infeasible Solution Bypass

Another method of dealing with constraints and their effects on the search space is the implementation of a bypass system in metaheuristics. This might involve using specialised operators in a GA that ensure offspring solutions are always feasible or designing the search process in a way that avoids constraint violations. This may accomplished by introducing a specific upper and lower bound for each variable, defined by the constraints, built into the mutation and crossover operators of a GA. Additionally, it is possible to design the algorithm such that any solution that violates a constraint is rejected before it is evaluated. One approach, developed in [42] and shown in Equation (2.5), shows a "feasibility first" penalty term used internally that builds upon the rejection of infeasible solutions. As noted by the authors, they aimed to create a new feasibility-aware method for population-based metaheuristics. As stated: "The pair-wise comparison used in tournament selection is exploited to make sure that: ..."

- When two feasible solutions are compared, the better solution is selected.
- When one feasible, one infeasible are compared only the feasible is selected.
- When two infeasible are compared, the smaller violating is selected.

$$F(\overrightarrow{x}) = \begin{cases} f(\overrightarrow{x}), & \text{if } g_j(\overrightarrow{x}) \geq 0, \forall j \in J \\ f_{max} + \sum_{j=1}^{J} \langle g_j(\overrightarrow{x}) \rangle, & otherwise \end{cases} \tag{2.5}$$

Here $f(\overrightarrow{x})$ is the objective function, the constraint violation of constraint j is shown as $\langle g_j(\overrightarrow{x}) \rangle$ and the fitness function is $F(\overrightarrow{x})$. In the equation, the number of constraints is denoted by J. As noted by the authors, in a feasible solution $F(\overrightarrow{x}) = f(\overrightarrow{x})$ which is observed when no constraints are violated ($g_j(\overrightarrow{x}) \geq 0, \forall j \in J$). When this condition is not met, the alternative method of calculating $F(\overrightarrow{x})$ is used. Lastly, f_{max} is the maximum fitness value of all the feasible solutions in the population being tested.

Repair Operators

Repair operators tend to be problem-specific and are used to correct an infeasible solution by altering its values such that it no longer violates the set of defined constraints. Shown in Algorithm 1 is PYMOO [43] example of a repair operator designed for the knapsack problem [44]. Here, the aim is to calculate the highest total value that can be obtained by selecting a subset of items that all fit into a "knapsack". Each item has a value and an associated cost, in this case weight. The knapsack has a weight limit so this cannot be violated.

Algorithm 1 Consider Maximum Weight Repair Strategy for Binary Knapsack Problem

```
procedure ConsiderMaximumWeightRepair($x$, $C$)
$x \leftarrow$ binary decision vector for the solution (length = number of items)
$C \leftarrow$ maximum capacity for the knapsack problem
Calculate weight $w$ for the solution based on $x$ and item weights
for each item $i$ in $x$ do
    $x_i \leftarrow x[i]$ {inclusion decision for item $i$}
    while $w > C$ do
        {Capacity violation check}Select an item $j$ randomly from $x$ where $x[j] = \text{True}$
        $x[j] \leftarrow$ False {Remove the item} Update $w$ to reflect the removal of item $j$
    end while
end for
return $x$
end procedure=0
```

In this repair operator, the values in a solution - in this case, the decision whether to include or not an item - are inspected. As the solution violates the hard constraint of the maximum capacity of the knapsack, the repair operator will attempt to fix the solution. In the algorithm, each approved item is randomly removed until a feasible solution is found. This operator would be used on all infeasible solutions in a generation.

2.2.3 Generalised Metaheuristic Search

As mentioned, metaheuristics represent a higher level of abstraction than simple heuristics, allowing them to provide a framework that can be applied to a large array of optimisation problem types. In this thesis we will generalise the processes found in metaheuristic search into three distinct steps. All trajectory-based metaheuristics - optimisation algorithms that improve upon an initial solution in their iterative search for the optimal - share three specific steps detailed below. This allows for the creation of a general understanding of how these algorithms navigate the search space, adapt based on learned information and transition towards improved solutions.

Evaluation Step

The evaluation step represents the first stage of any algorithm iteration. Common to all is the need for each of these selected solutions to be scored according to the problem's fitness function. In this step, feasible solutions in a population are scored according to the fitness function and checked against any constraints that may apply. This allows the following learning step to determine suitable candidates for use in the updating of the either implicit or explicit model representing the algorithm's gained knowledge.

Learning Step

The learning step can be considered as drawing from a stochastic sampling process that samples a new population from a probability distribution determined by the current population and other (possibly implicit) states of the algorithm. In essence, this step represents any EA's methods of knowledge extraction from the selected set of solutions. Because of this, the learning step is the most algorithm-specific of the three steps. During this step, candidate solutions for consideration are selected by the algorithm. This can take many forms such as fitness-proportional selection, randomly selected neighbours or determining particle movement direction. In Differential Evolution, this would take the form of the generation of a trial vector, as outlined later in this chapter (2.2.7). This step is designed to ensure that the best candidate solutions, as scored by the fitness function, have the highest chance of passing on their current gained knowledge to the next generation of solutions. Each algorithm achieves this through different means ranging from the selection, crossover and mutation operators found in GAs to the covariance matrix adaptation found in certain evolutionary strategies.

Update Step

In the update step, the given algorithm will transition from its current solution, population of solutions or position in the search space to the next. This facilitates its movement across the fitness landscape in search of higher-quality solutions. This can include updating the positions of particles in swarm intelligence methods, or iteratively refining a single solution in local search strategies. In population-based metaheuristics, the output of this step is always a successor population representing any newly acquired information.

2.2.4 Local Search - HillClimbers and Neighbourhood Search

Local search algorithms operate by exploring the neighbourhood structure of search spaces, moving from one candidate solution to a neighbouring one sequentially. The primary advantage of these methods is their simplicity and efficiency in converging to a solution on particular types of search spaces .

A sub-type of the local search algorithm is hill-climbing, which transitions from a candidate to a neighbouring candidate that displays higher fitness. Most heuristic approaches operate under the assumption that neighbouring candidates are likely to have a similar or better function value. For a hill-climber to ensure reaching the optimum solution in a single run, a monotonic fitness increment is required. This can be followed from a randomly chosen starting point to the global optimum. However, a simple hill-climbing search might fail to reach the global optimum if it encounters a local optimum. This occurs because the hill-climber, by design, cannot accept a degrading move. If it encounters a point that is locally better than those surrounding it, the algorithm will not be able to escape this, as doing so would require the acceptance of a lower quality solution to continue the search for the global optimum .

When compared to Variable Neighbourhood Search (VNS) [45, 46], this inability to escape a local optimum is due to the differing neighbourhood structures each approach uses. The hill climber's neighbourhood is the set of immediately surrounding solutions within a specified number of allowable changes, such as the 2-bit flip neighbourhood in binary search spaces. The size of a hill-climbers neighbourhood is dependent on the number of allowable changes per move and is not limited to only one direction. However, Variable Neighbourhood Search employs a strategy of systematically changing the neighbourhood structure, allowing it to explore a larger set of potential candidates and thus providing a mechanism to escape local optima, as illustrated in Figures 2.3 and 2.4 where the dimensionality is $n = 1$.

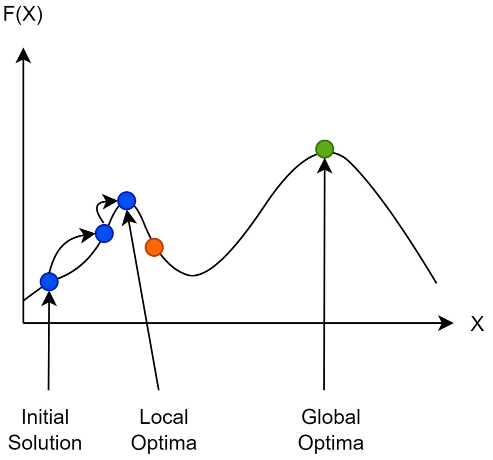


Fig. 2.3 Hill Climber

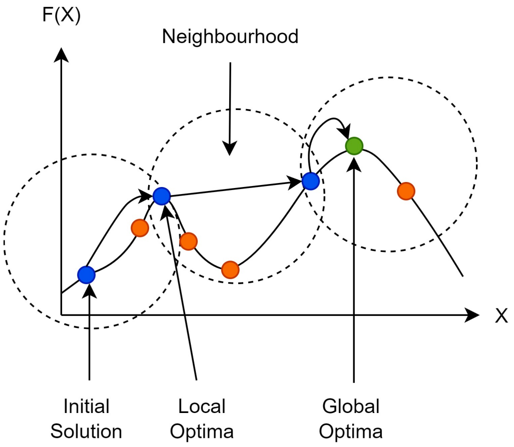


Fig. 2.4 Neighbourhood Search

In this example, we show how a Hill Climber will follow the higher fitness solutions until the next candidate solution is of poorer quality. We show how this may result in the algorithm finding a locally optimal solution but being unable to progress its search. These locally optimal solutions, along with those surrounding solutions from which a Hill Climber will inevitably climb to the local optimum are called the basis of attraction. Figure 2.4 shows how Variable Neighbourhood Search, with the use of multiple neighbourhoods of candidate solutions to choose from at each iteration, may escape these basins to find the globally optimal solution, provided that a higher quality solution is contained within the subsequent neighbourhood.

There exist many variations on local search metaheuristics, with two notable examples being Simulated Annealing and Tabu Search.

Simulated Annealing

Simulated annealing (SA) is a local search metaheuristic inspired by the physical attributes of thermodynamic free energy in a system as it cools [47]. Similar approaches to this have been independently developed previously [48, 49, 50]. This algorithm borrows terminology from physics, such as "Temperature," which controls a key aspect - the probability of accepting solutions that might not immediately improve the objective function. Here the variable allows degrading steps with a temperature-controlled probability that influences, over time, the decision on whether a solution is to be accepted. Shown in Equation (2.6a) is SA's ”acceptance function”. This function controls the probability that a currently considered solution will replace the best-found solution. This decision is based on the energy difference between the two (ΔH) as well as the current temperature (T) value.

$$P_{acc} = \exp\left(-\Delta H/T\right) \tag{2.6a}$$

$$T_{n+1} = \alpha T_n, 0 < \alpha < 1 \tag{2.6b}$$

This is used to calculate the value P_{acc}, the probability of accepting a move to the new position. Here, ΔH is the energy difference in which a negative value suggests an improvement in solution quality and a positive value suggests a reduction in quality. The temperature T, according to [51], is decreased via a schedule often defined as a geometric law in the form of Equation (2.6b) where α is the cooling rate, a constant value that is flexed between 0 and 1. This cooling rate flexing is a defining parameter of SA as seen in [52]. At higher temperatures (T), the probability of accepting a solution with a lower quality is still greater than zero. This allows the algorithm to escape local optima, where it might get stuck on suboptimal solutions. Simulated annealing typically starts with a randomly generated solution and iteratively creates new solutions through perturbation for comparison.

SA is an adaptation of the local search algorithm of Hill-Climbing with key internal changes. One significant adaptation, as noted by [53], is “The algorithm varies from Hill-Climbing in its decision of when to replace the original candidate solution with its newly tweaked child. Specifically: if the new solution is better than the original, we'll always replace the original. But if the new solution is worse, we may still replace it with a certain probability”.

This search strategy of SA helps differentiate it from the simpler Hill climber. During the search, the SA will normally select the higher quality neighbouring solutions much like the hill climber. When the temperature allows degrading behaviour, however, SA can escape local optima by selecting a poorer quality solution. The search trajectories of these algorithms, understood as the sequence of neighbouring solutions visited, will differ between algorithms because of this new behaviour. In essence, the SA will behave differently when it encounters a local optimum than hill climbing does, resulting in a different trajectory. In terms of understanding algorithm performance based on these trajectories, this differing behaviour is of great interest to this research.

Tabu Search

The Tabu search algorithm, introduced in 1986 [54], is designed to address the limitations of local neighbourhood search methods. It utilises a "tabu list" to temporarily restrict revisiting recently explored solutions, encouraging the algorithm to move away from any local optima it may have encountered. This behaviour also allows the Tabu search to visit areas of the search space that a traditional hill climber would not be able to access. Algorithm 2 outlines the core steps.

Additionally, aspiration criteria - a set of rules that allow solutions on the tabu list to be revisited - are another distinguishing feature of this search algorithm. An example of such criteria may be the restriction of revisiting a solution until a certain number of generations have passed, after which it is considered a viable option again or if it has a higher quality than previously found solution.

As can be seen from the pseudo-code, Tabu begins by initialising with a randomly selected solution in the space of all possible solutions S. The process requires a maximum number of iterations to run ($maxIter$) and objective function f. The immediate neighbourhood ($N(s,k)$) of points is then sampled ($V_{s,k}$) and a higher quality solution is sought. Movement to this new point is then allowed under two circumstances - the point is not in the Tabu list of off-limit solutions (T) and the aspiration criteria ($a(k)$) are met. A long-term memory list is then updated with the best found solution so far, allowing the algorithm to keep track of the highest quality solution visited over the whole optimisation run. Should the newly sampled point be of higher quality than the current, then the Tabu search will move to this new point and the process is repeated until all allowable iterations are completed. This search behaviour is different from the simpler hill climber in that, with the use of aspiration criteria

Algorithm 2 Tabu Search

```
procedure TabuSearch($S$, $maxIter$, $f$)
Initial solution selection $s \in S$
Initialize the best-so-far solution $s^* \in S$
$k \leftarrow 1$
while $k \leq maxIter$ do
    Sample the allowed neighbors of $s$
    Generate a sample $V(s,k)$ of the allowed solutions in $N(s,k)$
    $s' \leftarrow$ the best solution in $V(s,k)$ not in Tabu list
    $s' \in V(s,k) \Leftrightarrow (s' \notin T) \vee (a(k,s') = true)$
    if $f(s') < f(s^*)$ then
        $s^* \leftarrow s'$
    end if
    Update Tabu list and aspiration criteria
    $k \leftarrow k+1$
end while
return $s^*$
```

and memory lists, the algorithm is able to escape local optima. As a result, this behaviour will produce a different search trajectory from hill climbing as well SA.

2.2.5 Genetic Algorithms

Genetic Algorithms (GAs) are search metaheuristics based on the process of natural selection, where solutions to optimisation problems are evolved over time. In a GA, a solution is represented as a string or vector of symbols as a chromosome such as $x = [x_1, x_2, \ldots, x_n]$, and a population of these chromosomes is evolved to find better solutions to a given optimisation problem. First introduced in the 1960s and 1970s by Holland [11], they learn knowledge of the problem structure implicitly as each member of the current population generated represents the current understanding of the problem being solved. This means that the population and alteration methods (Selection, Crossover and Mutation) do not create an explicit model of the problem and instead the populations generated in each step of the process represent the algorithm's implicit understanding of the fitness function via members of that population.

As an example, we show the implementation of a rudimentary $(\mu + \lambda)$ GA, where μ is the number of parent solutions selected and λ is the number of the resulting solutions once the internal operators have been used. This algorithm generates each successive population of solutions by performing the Selection, Crossover and Mutation operations on the parent population. The main processes typically found in a $(\mu + \lambda)$ GA is shown in Algorithm 3.

Algorithm 3 $\mu + \lambda$ Genetic Algorithm

```
Initialize population with $\mu$ individuals
Evaluate the fitness of each individual in the population
while termination criteria not met do
    Select $\mu$ parents from the current population
    Generate $\lambda$ offspring using genetic operators
    Evaluate the fitness of each offspring
    Combine the $\mu$ parents and $\lambda$ offspring into a single pool
    Select the best $\mu$ individuals from this combined pool to form the new population
    Evaluate the fitness of the new population
end while
return The best individual from the final population =0
```

An important aspect of this type of GA to note is the difference between the $(\mu+\lambda)$ and (μ,λ) evolutionary strategies. As stated, the $(\mu+\lambda)$ selects the next generation of solutions from a pool of both parent and generated solutions however the (μ,λ) replaces entirely the parent generation μ with the child best solutions in λ. Both strategies have their own benefits and drawbacks. $(\mu+\lambda)$ has the ability to preserve the best found solution so far however in dynamic optimisation where the objective may change, outdated high quality solutions will be retained and may hinder the search. (μ,λ) focuses more on the exploration of the space and may be of benefit in multi-modal landscapes however it risks losing high quality solutions as all parents are replaced.

The following operators are not exclusive to GAs and are often employed in a variety of EAs in some form or another. The descriptions are aimed at providing an introduction to the fundamentals of their function and their often non-deterministic nature. As can be seen in later algorithm overviews, several EA variants have replaced or adapted these to suit their own search methodologies.

Selection

This is used to reduce the current population of solutions to a smaller population of suitable candidates based on metrics such as their fitness values. As noted in [55], the selection method can be classified as ordinal or proportional. These define the overall process of how parent solutions are selected. In ordinal selection, solutions are selected based on their fitness values via direct comparison. In proportional selection the fitness of a solution impacts the probability that it will be selected over a lower quality solution however this is not guaranteed. Table 2.1 shows some examples of selection methods and their category.

Selection Type	Method Name	Deterministic/Stochastic
Ordinal	Tournament	Stochastic
Ordinal	Truncation	Deterministic
Proportional	Boltzmann	Stochastic
Proportional	Roulette Wheel	Stochastic

Table 2.1 Summary of Genetic Algorithm Selection Methods

Tournament selection involves randomly selecting a subset of individuals from the population and then choosing the best individual from this subset to become a parent in the crossover phase. The process is repeated until enough parents are selected. The size of the tournament can vary, affecting the selection pressure. Smaller tournaments lead to less pressure and more genetic diversity, while larger tournaments increase the pressure and can speed up convergence [56].

Truncation selection involves selecting the top k best fitness solutions in the current population of solutions. This method is often used when the goal is to rapidly converge to a solution, though it can lead to premature convergence and loss of population diversity.

Equation (2.7) shows how a fitness-proportional Roulette selection method can be used to calculate the probability of a solution (p_i) to be selected.

$$p_i = \frac{f_i}{\Sigma_{j=1}^{N} f_j} \tag{2.7}$$

Where P_i is the probability of the solution being selected, f_i is its fitness value, N is the population size, and f_j is the fitness of the j^{th} member of the population.

Boltzmann selection, also known as Boltzmann tournament selection, is inspired by the principles of statistical mechanics. This method selects individuals based on a probability distribution related to their fitness and a temperature parameter that controls the selection pressure [47]. Equation (2.8) shows the definition of this selection method.

$$p_i = \frac{e^{\frac{f(i)}{T}}}{\Sigma_{j=1}^{N} e^{\frac{f(j)}{T}}} \tag{2.8}$$

Here, T is the temperature parameter and e is the base natural log. As the temperature cools, the probability of selecting a lower quality solution reduces.

Crossover

Crossover is considered to be the operator that most distinguishes a GA from other population based metaheuristics. This is the process of generating new members of the next population from those selected in the previous step by exchanging partial solutions, or sub-strings, to generate a new solution. Depending on how many crossover points are being used, this process breaks apart and recombines selected “parent” genes to generate children for the next population to overall increase the next population’s fitness score.

In [57], it is shown that the crossover operator can be categorised as either geometric or non-geometric, which establishes that many commonly used crossover operators can be characterised mathematically as selecting a child from a generalised convex combination of the parents. In this work, a crossover operator is geometric if it has three specific properties " . . . arising only from its axiomatic definition (metric axioms), hence valid for any distance, any probability distribution and any underlying solution representation" [57]. These properties are:

- Property of Purity - "If the operator RX is geometric then the recombination of one parent with itself can only produce the parent itself."
- Property of Convergence - "If the operator RX is geometric then the recombination of one parent cannot produce the other parent of that offspring unless the offspring and the second parent coincide."
- Property of Partition - "If the operator RX is geometric and **c** is a child of **a** and **b**, then the recombination of **a** with **c** and the recombination of **b** with **c** cannot produce a common grandchild **g** other than **c**."

Shown in Figure 2.5 is an adaptation of their diagram showcasing inbreeding and how these properties are visualised.

The authors note that these properties are adaptations of geometric interval spaces outlined in [58]. This adaptation is key as it allows for the definition of geometric crossover to cover all operators that function by generating offspring as long as they are in the line segment between their parents under any defined distance metric d. By approaching crossover in such a manner, this can generate a representation-independent method of categorising crossover operators.

Examples of geometric crossovers include masked crossovers such as single-point and multi-point crossover [59, 60] in binary string and other fixed-length vector representations. In single-point crossover, a crossover point is chosen, and the parts of two parent chromosomes beyond that point are swapped to create offspring. A variation of this method

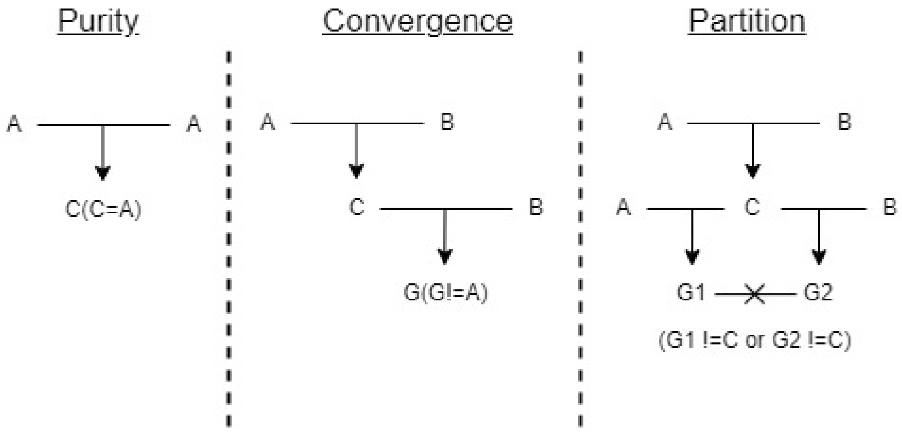


Fig. 2.5 Geometric Crossover Inbreeding Diagrams from [57]

is Multi-Point Crossover in which multiple fixed points are used across the chromosomes, and then the sub-strings or sub-vectors between each crossover point are swapped between solutions. Crossover can also take place on a gene by gene basis, as seen in the uniform crossover method, in which each gene is independently chosen from one of the parents with equal probability.

Mutation

This is the process of selecting, at random, child solutions of the selected parents and then altering one or more of their alleles to introduce variety to the population which tends to decrease as the population converges. This aims to model the process of random genetic variation that occurs between parents and their offspring in nature. After mutation, the offspring are included in the final population of solutions generated by the process. In binary string representations, bit-flip mutation is often used in EA to mutate the elements of a solution. Shown in Equation (2.9) is typically how bit-flip mutation will change a binary value to its corresponding opposite value.

$$x'_i = \begin{cases} 0 & \text{if } x_i = 1 \\ 1 & \text{if } x_i = 0 \end{cases} \tag{2.9}$$

This is performed at a set probability for each bit in a solution with values such as $1/n$ where n is the length of the bit string however it is not uncommon for different mutation rates to be used to increase or decrease the rate of mutation over the course of an optimisation run.

In real-valued vector representations, methods such as Gaussian and Polynomial mutation can be used. Both methods can use the same gene selection method as bit-flip, with a pre-defined probability of being selected for mutation. If selected, the value of the gene is edited according to the mutation operator's specific mechanism. In Gaussian mutation, the new value for the gene is calculated as shown in Equation (2.10).

$$x_i' = x_i + N(0, \sigma^2) \tag{2.10}$$

$$x_i' = x_i + \sigma \cdot (x_{max} - x_{min}) \tag{2.11}$$

Here, the original value is updated by adding a randomly generated value based on a normal or Gaussian distribution using the mean value of 0 and variance of σ^2. Similarly, as shown in Equation (2.11), Polynomial mutation uses a distribution (σ) to update the gene value. In this method, a value is selected from a bounded range and a polynomial distribution function. In [61] and utilized in the PYMOO framework used in this thesis, this polynomial mutation operator can be designed to use the same distribution function as found in the simulated binary crossover detailed in the same publication.

2.2.6 Estimation of Distribution Algorithms

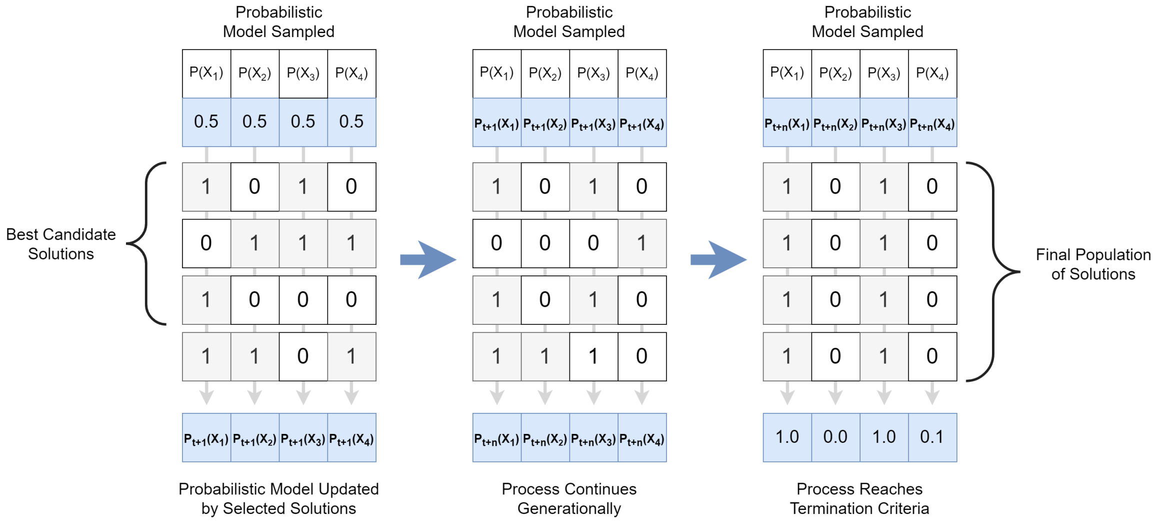

Probabilistic Model Sampled
P(X1) P(X2) P(X3) P(X4)
0.5 0.5 0.5 0.5
Best Candidate Solutions
1 0 1 0
0 1 1 1
1 0 0 0
1 1 0 1
Pt+1(X1) Pt+1(X2) Pt+1(X3) Pt+1(X4)
Probabilistic Model Updated by Selected Solutions
Probabilistic Model Sampled
P(X1) P(X2) P(X3) P(X4)
Pt+1(X1) Pt+1(X2) Pt+1(X3) Pt+1(X4)
1 0 1 0
0 0 0 1
1 0 1 0
1 1 1 0
Pt+n(X1) Pt+n(X2) Pt+n(X3) Pt+n(X4)
Process Continues Generationally
Probabilistic Model Sampled
P(X1) P(X2) P(X3) P(X4)
Pt+n(X1) Pt+n(X2) Pt+n(X3) Pt+n(X4)
1 0 1 0
1 0 1 0
1 0 1 0
1 0 1 0
1.0 0.0 1.0 0.1
Final Population of Solutions
Process Reaches Termination Criteria

Fig. 2.6 EDA Mechanisms Example

Estimation of Distribution Algorithms (EDAs) function by creating an explicit probabilistic model of what high quality solutions in the current population look like then samples that model to create a successor population [62]. Population Based Incremental Learning (PBIL) is an EDA that uses a univariate probabilistic model, meaning that each variable in the problem is considered independent and any co-dependencies between variables are not explicitly captured. The probability vector is updated and mutated each generation as seen in Equation (2.12) and (2.13):

$$p(X_1,\ldots,X_N) = \prod_{i=1}^{N} p(X_i) \tag{2.12}$$

$$p(X_i) = \frac{1}{N}\sum_{j=1}^{N} x_{ij} \tag{2.13}$$

Here the vector of marginal probabilities $P_V = (p(X_1),...,p(X_n))$ is created by calculating the arithmetic mean of each variable X in a population of size N. As the solutions are comprised of bit strings we will see that values will range from 0 to 1. Advancements in this area have included the introduction of Markov Random Fields [63] to improve on the performance and allow EDAs such as Distribution Estimation using Markov Random Fields (DEUM) [64] to solve higher order problems. The ability of EDAs to exploit the interaction between variables has also seen significant advances. Bi-Variate examples MIMIC [65] and beyond this, multivariate extensions exist such as the multivariate DEUM [66]. There are several Bayesian based implementations such as the Bayesian Optimisation Algorithm BOA [67], a hierarchical adaptation of BOA in hBOA [68] and EBNA [69]. Further notable works in this area include the development of the Recombinative Optimal Mixing Evolutionary Algorithm (ROMEA) and the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) in [70].

2.2.7 Differential Evolution

This method of optimisation involves the perturbation of a vector representation of populations of solutions using vector differences. This process of moving from one population of solutions to the next is outlined in Equation (2.14) and is based on the single objective optimisation algorithm set in [15].

Here v_i is the vector representing the donor vector that is created by the mutation and crossover process for individual solution *i*. x_{r1}, x_{r2} and x_{r3} are solutions in r, a collection of three randomly selected, mutually exclusive solutions in the current population. The F value is a scaling factor that takes a value between 0 and 1 and i used to alter the differential variation of $(x_{r2} - x_{r3})$. Once complete a second crossover event takes place between a given individual solution and the donor vector v.

$$v_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3}) \tag{2.14}$$

The crossover operation in DE combines the mutant vector v_i with the target vector x_i to create a trial vector u_i. This is shown in Equation (2.15).

$$u_{ij} = \begin{cases} v_{ij} & \text{if } \text{rand}_j \leq CR \\ x_{ij} & \text{otherwise} \end{cases} \tag{2.15}$$

In the crossover equation, $u_{i,j}$ represents the j-th component of the trial vector. $rand_j$ is a uniformly distributed value between 0 and 1 and CR is the crossover rate, also a value between 0 and 1.

$$x_i^{g+1} = \begin{cases} u_i & \text{if } f(u_i) \leq f(x_i) \\ x_i & \text{otherwise} \end{cases} \tag{2.16}$$

After this has occurred, the selection process is performed to compare the target and trial vectors created based on their fitness. Whichever has the better fitness value is selected for the next generation, as shown in Equation (2.16). Here, g is the generation number at which the process is taking place. An illustration of these, from [71], can be seen in Figures 2.7, 2.8 and 2.9

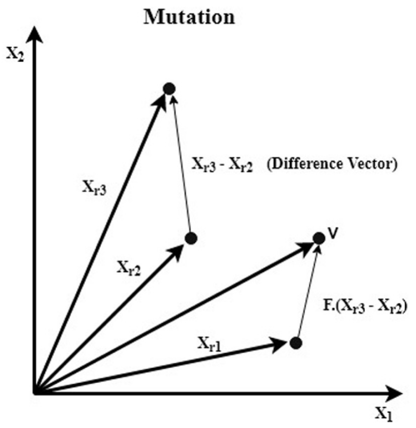


Fig. 2.7 Differential Evolution Mutation from [71]

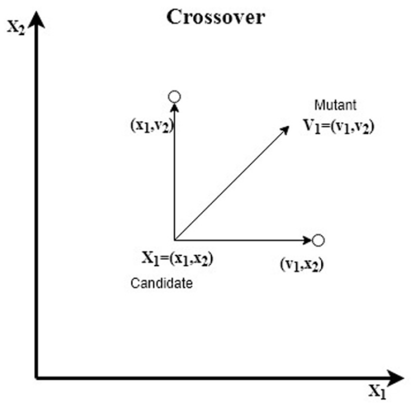


Fig. 2.8 Differential Evolution Crossover from [71]

Since its inception in 1996 [72], there have been several iterations of development that have built upon the base DE algorithm. These include the adaptation to include fuzzy logic controllers to update the search parameters for the mutation and crossover operators [73]. Adaptations include the exploitation of neighbourhood structures seen in the Neighbourhood Based Differential Evolution (NDE) [74] and the use of multiple trial vectors per solution

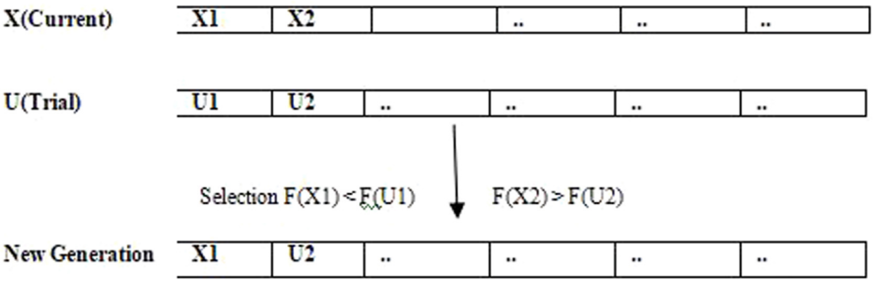


Fig. 2.9 Differential Evolution Selection from [71]

Algorithm 4 Differential Evolution Algorithm

```
Initialize a population of solutions
while termination criteria not met do
    For each target vector, generate a trial vector by mutation and crossover
    Perform selection to generate the next generation
end while=0
```

as seen in the Composite Differential Evolution (CoDE) algorithm [75]. The use of self-adaptation can be seen in [76], developed in 2007, this novel method allows DE to exploit a new mutation strategy, leading to the "‘DE/current-top-best’ variant of DE allowing for a more diverse population but maintaining the quick convergence performance. Building on this, the SHADE [77] variant of DE has been developed. It aimed to create a new parameter adaptation mechanism that is "...based on a historical record of successful parameter settings." and shows competitive results against other state-of-the-art DE algorithms.

2.2.8 Particle Swarm Optimisation

This involves the use of multiple particles that each represent a candidate solution and tracking their positional change through the search space. Their search is controlled through the application of a calculated velocity and direction that is dependent on the swarm’s current best solution and neighbouring particle’s performance. The implementation of the PSO is based on [78] with alterations to allow the values for c_1 and c_2 to be updated at the same time. This alteration is based on the implementation found in [79]. Equation (2.17) shows the formula for how each particle’s velocity is updated between generations

$$V_d^{(i)} = \omega V_d^{(i)} + c_1 r_1 \left(P_d^{(i)} - X_d^{(i)} \right) + c_2 r_2 \left(G_d^{(i)} - X_d^{(i)} \right) \tag{2.17}$$

$$X_d^{(i)} = X_d^{(i)} + V_d^{(i)} \tag{2.18}$$

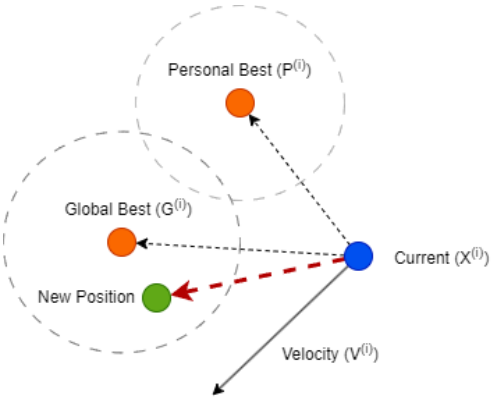


Fig. 2.10 PSO Update

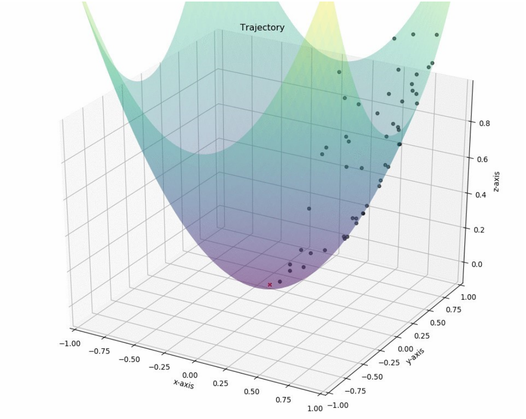


Fig. 2.11 PSO Search Space

$X_d^{(i)}$ represents the d-th coordinate of i-th particle's position. $V_d^{(i)}$ is the d-th coordinate of i-th particle's velocity. ω the the weight applied to the inertia values. $P_d^{(i)}$ represents the d-th coordinate of i-th particle's personal best fitness value. $G_d^{(i)}$ d-th coordinate of the global best solution found so far. This value can also represent the local best solution should they be the same. c_1 and c_2 are the to weight values used in order to balance exploiting the particle's best, $P_d^{(i)}$, and swarm's best $G_d^{(i)}$. Finally r_1 and r_2 are used to represent the two random values used in the creation of the velocity update.

Algorithm 5 Particle Swarm Optimisation Algorithm

```
Initialize a swarm of particles
while termination criteria not met do
    Update the velocity and position of each particle
    Update the personal and global best positions
end while=0
```

Equation (2.18) displays how the position of each particle is also updated between populations. Here, P is the fixed population size, n is the number of dimensions in the problem and *runs* is the total number of optimisation runs in the trajectory.

2.2.9 Covariance Matrix Adaptation Evolution Strategy

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [80] is a form of evolutionary algorithm designed to be a stochastic derivative-free numerical optimisation method for non-linear or non-convex continuous optimisation problems including often difficult problems such as those with rugged fitness landscapes. Equation (2.19) outlines how a population of solutions may be sampled for any given generation.

$$X_k^{(g+1)} \sim N(m^{(g)}, (\sigma^{(g)})^2 C^{(g)}) \text{ for } k = 1, \ldots, \lambda \tag{2.19}$$

- $\sim$ denotes the same distribution on both sides.
- $N(m^{(g)}, (\sigma^{(g)})^2 C^{(g)}) \sim m^{(g)} + \sigma^{(g)} N(0, C^{(g)}) \sim m^{(g)} + \sigma^{(g)} B^{(g)} D^{(g)} N(0, I)$ is the multivariate normal search distribution.
- $X_k^{(g+1)} \in \mathbb{R}^n$ is the k-th offspring from generation $g+1$
- $m^{(g)} \in \mathbb{R}^n$ the mean value of the search distribution at generation g.
- $\sigma^{(g)} \in \mathbb{R}_+$ the overall standard deviation at generation g.
- $C^{(g)} \in \mathbb{R}^{nxn}$ is the covariance matrix at generation g.
- $\lambda \geq 2$ is the population size.

The main steps involved in generating a population of solutions using CMA-ES are shown in Algorithm 6. At each iteration, a new set of solutions is sampled from the multi-variate normal distribution according to the mean values ($m^{(g)}$) and covariance $C^{(g)}$ matrix. The population of solutions are then evaluated by the fitness function and the best performing are selected. These solutions are used to update the internal parameters of the algorithm, being the mean values, covariance matrix and the step-size or standard deviation $\sigma^{(g)}$ value at that generation.

Algorithm 6 CMA-ES Algorithm

```
Initialize a population of solutions, covariance matrix, and step-size
while termination criteria not met do
    Sample new solutions from a multivariate normal distribution
    Update the covariance matrix and step-size based on the fitness of solutions
end while=0
```

2.2.10 Stochastic Internal Operators and Non-Determinism

EAs have long been recognized for their robustness and adaptability in exploring solution spaces for a wide range of optimisation problems. However, as these algorithms have evolved to cater to diverse search methodologies, their operators have undergone significant adaptations, adding layers of complexity. This evolution, while enhancing the algorithms' capabilities, has also made them less tractable for non-experts, particularly in understanding their mechanisms and search behaviours. Shown in Table 2.2 is a summation of the evaluation, learning and update step mechanisms specific to each of the discussed EA types.

Algorithm	Evaluation	Learning	Update	Type
GA	Fitness Function	Selection, Crossover, Mutation	Population Replacement	Vector
EDA	Fitness Function	Selection, Probability Model Update	Probability Model Sampling	Vector
PSO	Fitness Function	Best-Found Update Position and Velocity Update	Update Personal and Global Best	History / Vector
CMA-ES	Fitness Function	Multivariate Model Update, Covariance Matrix and Step-Size Update	Multivariate Model Sampling	History / Vector
DE	Fitness Function	Trial Vector Creation, Crossover, Mutation	Vector Replacements	Vector

Table 2.2 Algorithm Step Overview and Type

Here, we show that a rudimentary GA utilizes the Selection, Crossover and Mutation operators. In contrast, EDAs have replaced the crossover and mutation stages, focusing instead on building probabilistic models of promising solutions [16]. This approach represents a distinct change of direction from the original GAs, emphasizing the learning of solution distributions over random explorations through genetic operators.

PSO algorithms and CMA-ES represent a further change in design from the GA by removing the traditional EA operators like selection, crossover, or mutation. Instead, these algorithms rely on unique operators such as swarm intelligence and adaptive strategies. PSO, for example, updates the trajectories of individual solutions based on social and cognitive behaviours. CMA-ES, on the other hand, adapts its search strategy by learning the covariance matrix of successful solutions, thereby efficiently adapting to the shape of the problem's fitness landscape [81].

While these diverse mechanisms show the adaptability and effectiveness of EAs, they also make them seem like black-box techniques. The details of these operators and how they interact within each algorithm can be hard for non-experts to understand. The non-deterministic nature of these algorithms means that understanding them fully requires not only knowledge of their basic principles and mechanisms but also an understanding of probabilistic and statistical concepts. This can be summarised in three specific characteristics of the algorithms - Their stochastic internal processes, the adaptive nature of their search strategies and the exploitation of probability models for decision-making.

However, all these algorithms have one thing in common: they are population-based. This means they create new generations of solutions, which provide a way to analyse and explain their processes. In later chapters, we will look at how this common feature can be used to understand these algorithms better. By examining how populations evolve, we can learn more about the decision-making processes in these algorithms, offering some insight into their otherwise black-box workings.

2.2.11 Metaheuristic Use in Industry

One of the primary motivations for using population-based metaheuristics in industry is the need to solve complex optimisation problems where traditional methods fall short, either due to the problem's complexity or the need for faster, near-optimal solutions. In many industrial applications, the use of computationally expensive parameter tuning is not available due to time constraints and the dynamic nature of industrial problems. This limitation often results in the implementation and use of algorithms that can perform reasonably well with default or minimal parameter tuning. For instance, in manufacturing process optimisation, where time is a critical factor, GAs have been used effectively with limited parameter adjustments [82]. Similarly, when parameter tuning has been shown to improve overall algorithm performance, it has also been noted that running these algorithms with the default parameter settings " ...is a reasonable and justified choice, whereas parameter tuning is a long and expensive process that might or might not pay off in the end" [83]. In an industrial setting where time is a constraint, a simplified version of the problem or algorithm implementation with little to no parameter tuning may be preferred.

Given the constraints on run time and the expertise required for parameter tuning, industries often prefer simpler or less complex algorithms with fixed parameters. These algorithms, despite their relative simplicity, are still complex non-deterministic algorithms and solvers. In current industry practice, most work involving EAs focuses on simplified versions of problems and the algorithms themselves. This approach aims to gain insights into the problem definitions and algorithm behaviours in a more controlled and understandable manner. Despite the simplifications, the need for explanations of how these algorithms arrive at solutions remains critical. In an industrial context, an explanation might involve demonstrating how the algorithm navigates the search space, how population diversity affects solution quality, or how the algorithm's operators contribute to finding the optimal solution.

Transport and Logistics

EAs are instrumental in optimising transport and logistics operations. An extensive overview of their applications in this field can be found in [23]. Whether it's optimising airport ground vehicle transport and taxi time, as demonstrated in [84] and [85], or addressing dynamic scheduling challenges like roster allocation and refinement under uncertain demand profiles, as explored in [86], EAs have been shown capable of enhancing efficiency, reducing delays, and improving overall system performance. In these works, it has been shown that some of the selected algorithms were able to generate schedules in which "significantly fewer number of aircraft stops has been achieved by the CCP-QPPTW, implying that the allocated routes would probably save much fuel consumption at the same time as stop-and-go has been identified as one of the main sources towards increased fuel burn" [84]. These results show increasing relevance as industrial fuel economy and emissions grow in the public interest. In terms of scheduling, shown in [86], the trialled EAs were able to produce a higher-quality set of staff rotas which took into account "uneven shift demand patterns while also satisfying multiple staff preferences related to their work-life balance.". This ability to solve not just for the most efficient rotas, but also generate alternative solutions in which non-specified goals such as work-life balance are improved shows the benefits that EAs can have in these areas. System-side improvements in transport systems can also be achieved EA as seen in [24], further highlighting their potential.

Engineering and Technology

The engineering and technology sector has seen growth in the design and implementation of EA-based solutions. Research in the area of big data and distributed computing using systems such as Spark [87] and Hadoop have become an active area. A demonstration of this can be seen in [88], showing their applicability in large-scale optimisation tasks, particularly in big data environments. Encoding techniques designed for separable large-scale multi-objective problems, as outlined in [27], contribute to optimising housing stock improvements and supporting urban planning and development. Additionally, the implementation of low-cost Internet of Things (IoT) cyber-physical systems, such as vehicle and pedestrian tracking in a smart campus, as described in [89], showcases EAs' relevance in modern technological advancements. Here, the mapping of students' movements was able to highlight the issue of foot traffic congestion in specific areas of a university, leading to higher noise load. Solutions taken on board include methods to alter the "spatio-temporal behaviour of the university community members" to reduce noise pollution. The work in [90] compares the output of two types of GAs - Traditional and Steady-State, as well as the effects of differing crossover operators on satellite-time schedules. This work aims to create a problem generator that closely matches the real-world issue of satellite usage scheduling for algorithm comparison.

Healthcare

Healthcare has also seen a rise in the application of many EAs and population-based metaheuristics which have been shown to play a vital role in offering innovative solutions to complex medical challenges. GAs have been shown to generate significantly better ambulance allocations during worst-case, peak utilisation scenarios using data for Oslo and Akershus [25]. Here, they show that the GA outperforms several other EAs and methods currently used and "optimizing dynamically on a weekly basis compared to a statical location based on the surrounding population is much more significant in the worst-case scenario". The multi-objective evolutionary design of antibiotic treatments, as explored in [91], aids in combating antibiotic-resistant pathogens by identifying optimal treatment regimens. In terms of real-world impact, this work and other similar studies have the potential to be game-changing in the fight against antibiotic-resistant pathogens, a growing concern in medicine across the world. In diabetic care, automated blood glucose regulation for nonlinear models of type-1 diabetic patients under uncertainties, as presented in [92], contributes to personalized treatment strategies and improved patient outcomes. The ability for a patient to have personalised treatments not just for short-term improvements, but similar technologies can be used for long-term customisation of drug administrations. Feature selection for disease diagnosis and enhancing diagnostic accuracy, as highlighted in the systematic review in [93] has also been explored. Many works also draw from earlier work in this area based on the Multi-objective optimisation of cancer chemotherapy, as discussed in [94], with the potential benefit being a significantly lower impact on quality of life while maximising treatment efficacy against tumours.

Energy

The energy sector benefits from EAs for optimizing renewable energy solutions and management systems. Multi-objective optimisation strategies for home energy management systems [95] showcase their role in sustainable home energy solutions, maximizing the utilization of renewable energy sources. In [96], EAs contribute to the efficient utilization of renewable energy sources, underlining their significance in achieving energy sustainability and reducing environmental impact. Further renewable energy examples of EA usage include the optimisation of land-based wind farm layouts, specifically in difficult and mountainous regions [28]. In this work, they use GAs and a "…capacitated minimum spanning tree approximation" to generate alternative layouts for turbine and cable layouts to maximise energy generation.

Given the impact that these algorithms have been shown to have in each of these areas, it is clear that the need for the generation of explanations is growing. With the continued growth of these technologies in many areas of modern society and industry, so does the need for explanations concerning their decision-making processes. This not only empowers the decision-makers that use these systems but also helps to justify key decisions and promote the benefits of EAs and other optimisation techniques which have the potential to transform many areas of modern life.

2.3 Explainable Artificial Intelligence

2.3.1 Overview

Explainable Artificial Intelligence (XAI) has rapidly emerged as a pivotal area of research and application, especially in the context of the increasing adoption of AI across various sectors. XAI aims to help improve the acceptance and trust in the output of modern AI systems - the black-box nature of these systems, where the decision-making processes are often opaque, complex, and difficult for humans to understand. A popular reference point for XAI is the Defense Advanced Research Projects Agency's (DARPA) XAI program "DARPA-BAA-16-53", based on the results of a systematic survey of AI researchers in 2016 [97]. From this, two significant areas were identified as being of high importance to the project:

1. Machine learning problems to classify events of interest in heterogeneous, multimedia data.

2. Machine learning problems to construct decision policies for a simulated autonomous system.

These issues were specifically selected as they allowed the project to focus, as shown in Figure 2.12, the intersection of two machine learning approaches (classification and reinforcement learning) and two important operational problem areas for the Department of Defence (DoD).

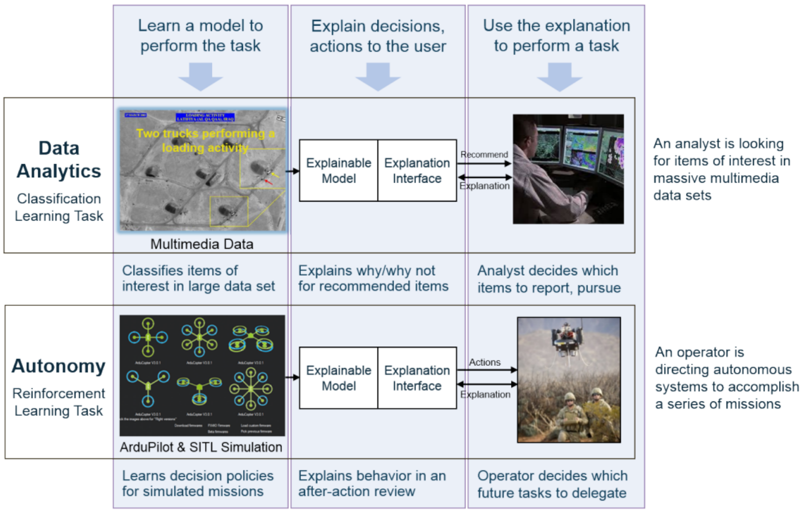

Learn a model to perform the task
Explain decisions, actions to the user
Use the explanation to perform a task
Data Analytics
Classification Learning Task
Two trucks performing a loading activity
Multimedia Data
Explainable Model
Explanation Interface
Recommend
Explanation
An analyst is looking for items of interest in massive multimedia data sets
Classifies items of interest in large data set
Explains why/why not for recommended items
Analyst decides which items to report, pursue
Autonomy
Reinforcement Learning Task
ArduPilot & SITL Simulation
Explainable Model
Explanation Interface
Actions
Explanation
An operator is directing autonomous systems to accomplish a series of missions
Learns decision policies for simulated missions
Explains behavior in an after-action review
Operator decides which future tasks to delegate

Fig. 2.12 DARPA XAI Challenge [97]

The initiative seeks to address the challenge of "Black Box" AI systems, where the decision-making processes are difficult to interpret or explain, which can be a significant concern in applications where transparency and accountability are paramount. DARPA aims to develop XAI technologies that provide clear and comprehensible explanations for AI system decisions, enabling users, operators, and decision-makers to understand why AI systems make specific choices or recommendations. In 2019, a new approach to develop AI techniques was suggested, based on the developments made in the preceding years. This new approach reflected the findings that there was often a trade-off between the performance of the techniques and the level of explainability of any generated explanations from the system. Shown in Figure 2.13 is the 2019 DARPA [98] general classification of different machine learning methods. In this, they also outline the perceived trade-off between explainability and predictive performance.

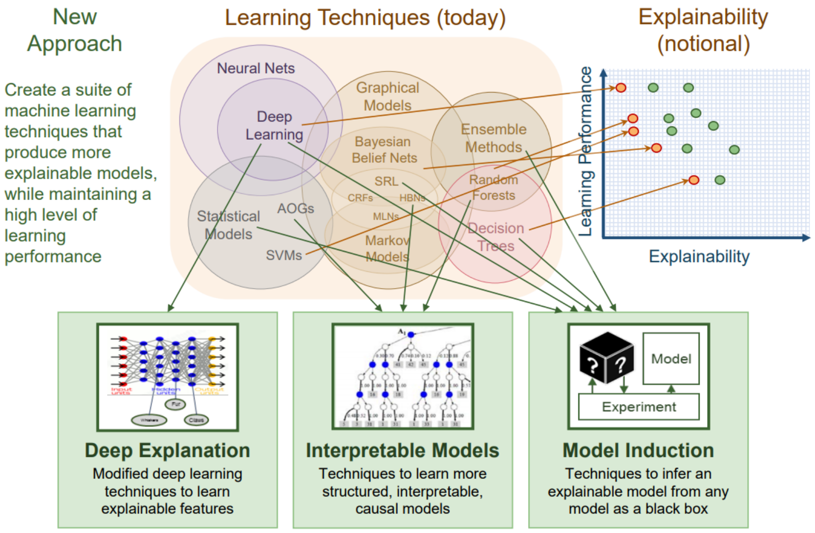

New Approach
Create a suite of machine learning techniques that produce more explainable models, while maintaining a high level of learning performance
Learning Techniques (today)
Explainability (notional)
Neural Nets
Deep Learning
Graphical Models
Bayesian Belief Nets
Ensemble Methods
SRL
CRFs
HBNs
MLNs
Random Forests
Statistical Models
AOGs
SVMs
Markov Models
Decision Trees
Learning Performance
Explainability
Model
Experiment
Deep Explanation
Modified deep learning techniques to learn explainable features
Interpretable Models
Techniques to learn more structured, interpretable, causal models
Model Induction
Techniques to infer an explainable model from any model as a black box

Fig. 2.13 DARPA XAI Overview [98]

Responsible AI encompasses the social, moral, ethical, and legal considerations surrounding AI systems, particularly those generating explanations from complex black-box models [99]. The goal is to enhance user understanding of problems and solutions. The field of Responsible AI emphasizes the A.R.T. principles for AI [100], which propose three pillars:

- Accountability refers to the need to explain and justify one's decisions and actions to its partners, users and others with whom the system interacts
- Responsibility refers to the role of people themselves and to the capability of AI systems to answer for their decision and identify errors or unexpected results
- Transparency refers to the need to describe, inspect and reproduce the mechanisms through which AI systems make decisions and learn to adapt to their environment, and to the governance of the data used and created.

These principles are echoed in the recommendations from a joint initiative by the PHG Foundation, the Information Commissioner's Office (ICO), and the Alan Turing Institute [101, 102, 103]. This initiative, driven by the increasing use of non-deterministic and "black-box" algorithms in medicine [104], explored topics including whether the argument of interpretability vs accuracy still holds true, evaluation methods and legal and ethical frameworks. The A.R.T. principles, echoing these recommendations, outline core ICO principles for AI:

- Be transparent.
- Be accountable.
- Consider the context you are operating in.
- Reflect on the impact of your AI system on the individuals affected, as well as wider society.

Over time, these principles have seen many iterations depending on the study or survey paper. Table 2.3 summarises the overlapping concepts extracted from our survey of the explainability literature. While not exhaustive, we rely on this base to derive working definitions of core terms sufficient for the purposes of this thesis.

Term	Definition
Understandability	This refers to the design of the internal mechanisms of a model, with the aim that they are readily understood by a user regardless of their true complexity.
Comprehensibility	In ML, this refers to the ability of a model to represent its learned knowledge of the problem in a human-understandable way.
Interpretability	This refers to a model's ability to represent and convey its findings in a user-type-specific language.
Explainability	Explainability refers to the interface that exists between the model and the user. This interface must be able to clearly and plainly explain the model's output while remaining accurate to the underlying data and is often considered a trade-off.

Table 2.3 XAI Concept Summary

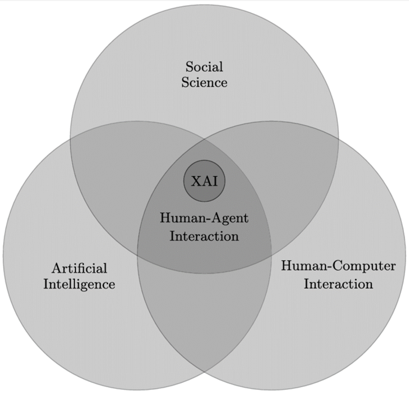


Fig. 2.14 Millers' Explainable Artificial Intelligence Concepts [105]

Interpretability Vs. Explainability

Frequently in the literature surrounding XAI, we find that two of the concepts - Interpretability and Explainability, are used interchangeably. Defining what an explanation is can be a difficult task, as noted by Miller [105], this process is subjective and can depend on the perspective of the subject. They point out that some consider an explanation to refer to the causes of the solution [106, 107, 108, 109], non-causal explanations [110] and the meaning behind a remark [111]. Miller's approach to the definition of XAI considers the area of research as a human-agent interaction and as such the social science aspect is considered a core point, as seen in Figure 2.14.

Lipton [112] challenges the notion of a one-size-fits-all approach to interpretability. He argues that interpretability is a spectrum, with different models offering varying degrees of transparency. This perspective is significant in highlighting the inherent trade-offs between model complexity and interpretability. Lipton's work suggests that in some cases, a more complex, less interpretable model might be necessary to achieve high levels of accuracy, while in other scenarios, a simpler, more interpretable model could be more beneficial.

The interpretability of models and AI systems is key to achieving this. As noted by [113], this can be difficult to define but can be understood broadly as the "extraction of relevant knowledge from a machine-learning model concerning relationships either contained in data or learned by the model" which also applies to metaheuristics.

In this thesis, we distinguish between interpretability and explainability using these definitions however we must also remember, as noted by [114] that both are related and required to understand a model [112, 115]. This closely aligns with their definitions of both terms in which interpretability "... focuses on the interpretability of the model and aims to improve it by simplifying or otherwise restricting its complexity" [116] and explainability "... aims to provide explanations by analyzing the model after it has been trained.". For the remainder of this thesis, when referring to an explanation, we are attempting to answer the question of why an observed algorithm behaviour has taken place, and how this contributes to its overall search for the same outcome - the (ideally) optimal solution.

To this end, we will use Miller's interpretation of an explanation to be "... an answer to a why–question". This follows their counterfactual reasoning that an explanation can be arrived at by "simulating alternative causes to see whether the event still happens". This definition was selected as it aligns well with this research - explanations generated from the features mined from algorithm search trajectories - in essence, using populations of alternative causes to generate an explanation regarding whether and why an optimal solution is consistently arrived at by the algorithm.

Global and Local Explanation

Local explanation in XAI refers to explanations that are specific to individual predictions or solutions generated by the system. It focuses on why the model made a particular decision for a single data point. In contrast, Global explanations aim to generate an explanation to increase a user's overall understanding of the model's behaviour, across all solutions. When generating local explanations, there are many possible techniques to select from. As an example, methods like LIME (Local Interpretable Model-agnostic Explanations) [117] and SHAP (SHapley Additive exPlanations) [118] are commonly used for local interpretable explanations. LIME has been designed to create an interpretable model around individual predictions to explain them.

Stakeholders and Use-Cases

Identifying stakeholders in the context of XAI is crucial for several reasons. As highlighted in a study by Hoffman [119] in Frontiers in Computer Science, stakeholders' needs vary significantly, necessitating tailored explanations of AI systems.

Stakeholders	Description
Developers	Build AI applications primarily to solve complex problems, implement intelligent systems, debugging, evaluation, and improving application robustness.
Theorists	Advance AI theory, delving into artificial neuroscience to understand deep neural networks, often overlapping with the developer community.
Data Scientists	Need comprehensive understanding of AI systems, including data, implemented models, predictions, and addressing system errors.
Users	Utilize AI systems, requiring explanations for informed actions based on system outputs, e.g., insurance companies using AI for policy decisions.
Consumers	Recipients of products/services, needing simple, clear explanations to use AI systems independently, enhancing trust and transparency.
Businesses	Seek to deploy AI within their product, including diverse stakeholders like policemen, judges, who need understanding of model decisions for fairness and protection from false decisions.
Regulators	Monitor and audit AI systems, ensuring models are up to date and tracing decision trails in case of false outputs.

Table 2.4 Description of Stakeholders in AI Development

Researchers have emphasized the importance of considering different stakeholder groups, as their requirements for explanations differ based on their roles, circumstances, and responsibilities. Domain-specific attempts at explanation definitions such as Health Care, Data Analytics, AI, and Verification and Validation [120, 121, 122, 123] are also highlighted in the paper. Shown in Table 2.4 is our interpretation of what the main stakeholders are in the area of AI development, based on the findings of [124]. The table also gives a short description of the main tasks that each of the stakeholders tends to perform. From this information, it was decided to focus on the two most likely candidates of stakeholders for our work. These are the User and the Developer. For the User, a more interpretable and simplistic explanation is the main focus. In the context of our work, this could take the form of highlighting specific variables that are key to certain solutions. This informs the User of the main drivers towards solution quality while remaining easily interpretable and informative to decision-makers. The Developer was selected as they often require a deeper level of understanding of not just the results, but the system itself. We aim to also generate some explanatory features capable of helping the Developer identify and compare algorithm search behaviours. Examples of user-specific ”Why?" questions and others that the techniques in this thesis aim to answer may include:

For the User

- Why does altering a variable alter the quality of the solutions found?
- Why did the system recommend one solution over another?
- Are there alterations to a solution I can make that does not make significant impact on output?

For the Developer

- Why does one algorithm perform differently on the same problem as others - are there aspects of the problem it is not detecting?
- Why do algorithms with similar or differing search methods perform differently on the same problem?
- Why does this algorithm perform well on one problem and poorly on another?

Explanations such as answers to these questions can be used to help these end-users make better-informed decisions. By providing insights into the decision-making processes of the AI systems, it can help build trust - especially in the case of the User. For the User, understanding the variables that drive solutions can lead to better strategic decisions, such as reallocating resources or adjusting objectives based on what influences outcomes the most. For the Developer, detailed explanations help in troubleshooting the system. Knowing how different algorithms behave and why one might be more effective than another allows Developers to refine these algorithms or choose the right approach for specific problems.

2.3.2 Taxonomies of XAI Research

The field of XAI is characterized by a diverse range of taxonomies, each reflecting different perspectives and priorities in the research community. These not only categorize the approaches and methods in XAI but also highlight the evolving understanding of what constitutes effective and necessary explanations in AI systems.

Prominent earlier work can be seen in the user-centred approach found in Doshi-Velez and Kim's study on the interpretability of machine learning [125]. Here, conclusions include the need to "...categorize our applications and methods with a common taxonomy ..." to help clarify where research falls in terms of applicability and technical approaches, specifically "Creating a shared language around such factors is essential not only to evaluation but also for the citation and comparison of related work.". Their work categorizes explanations based on whether they aim to improve model transparency or to enhance trust and persuasiveness. This distinction is crucial in understanding that different applications and user groups may have varying requirements for explainability. For instance, a healthcare professional might need detailed insights into a model's decision-making process, while a layperson may require a simpler, more intuitive explanation.

Gilpin et al. [126] provide a taxonomy that categorizes XAI approaches based on the type of explanation they offer. In this work, they focus on three specific areas of explanation generation - The processing of the data by the system, the representation of the data inside the system and the use of self-explaining architecture, more commonly referred to today as transparent models. This taxonomy is particularly useful in understanding the wide array of techniques available for making AI systems more interpretable. They conclude that the field of explainability often focuses on advancing specific categories of techniques, while relatively less emphasis is placed on integrating various categories to achieve more comprehensive and effective explanations. The combination of techniques and efforts from other fields is crucial to our ability to develop "...methods that provide behavioural extrapolation, build trust in deep learning systems, and provide usable insight into deep network operation enabling system behaviour understanding and improvement".

This drive for a more integrated, holistic view of the technologies and methodologies was later seen in Arrieta et al [127], which proposes a holistic taxonomy that emphasizes the integration of explainability throughout the AI system life-cycle. This approach is significant in recognizing that explainability is not just a post-hoc addition, but can also be split into model specific and model agnostic works. The survey paper also provides an overview of a significant portion of the XAI landscape. Included in their work is a breakdown of what they consider to be a model's level of explainability. Their work underscores the importance of considering explainability at every stage, ensuring that AI systems are not only effective but also transparent and accountable. A consolidated extract of the taxonomic structure developed for their survey paper can be seen in Figure 2.15.

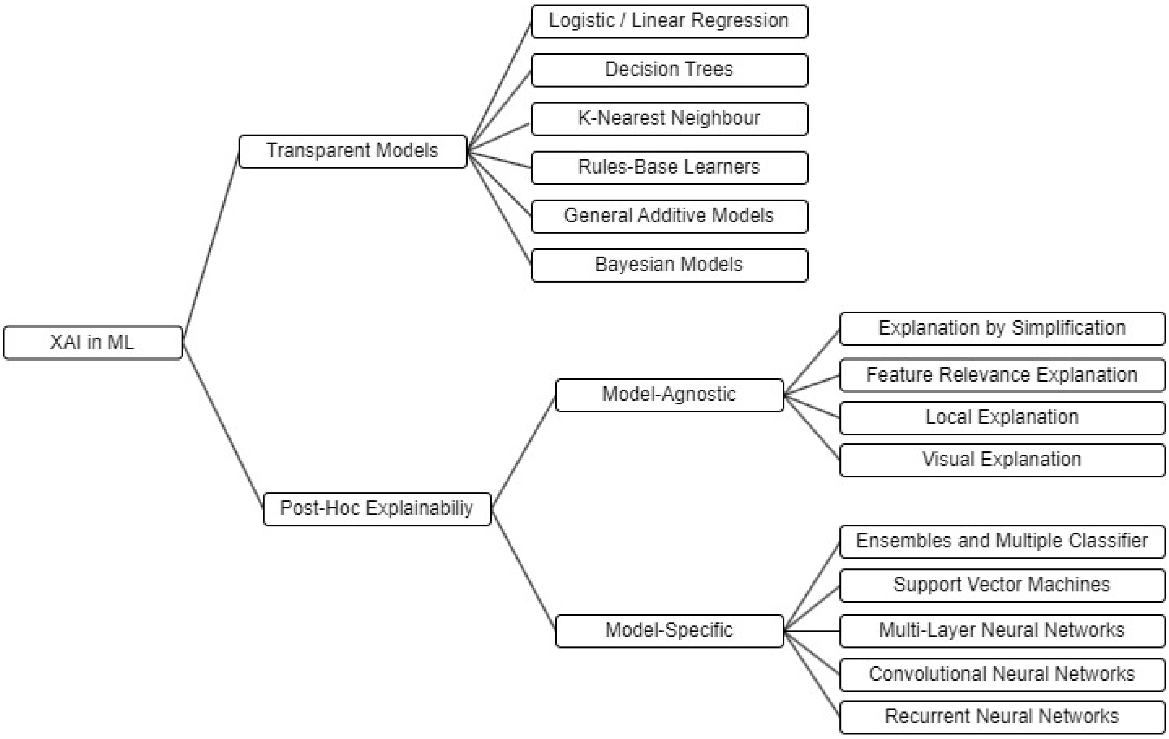
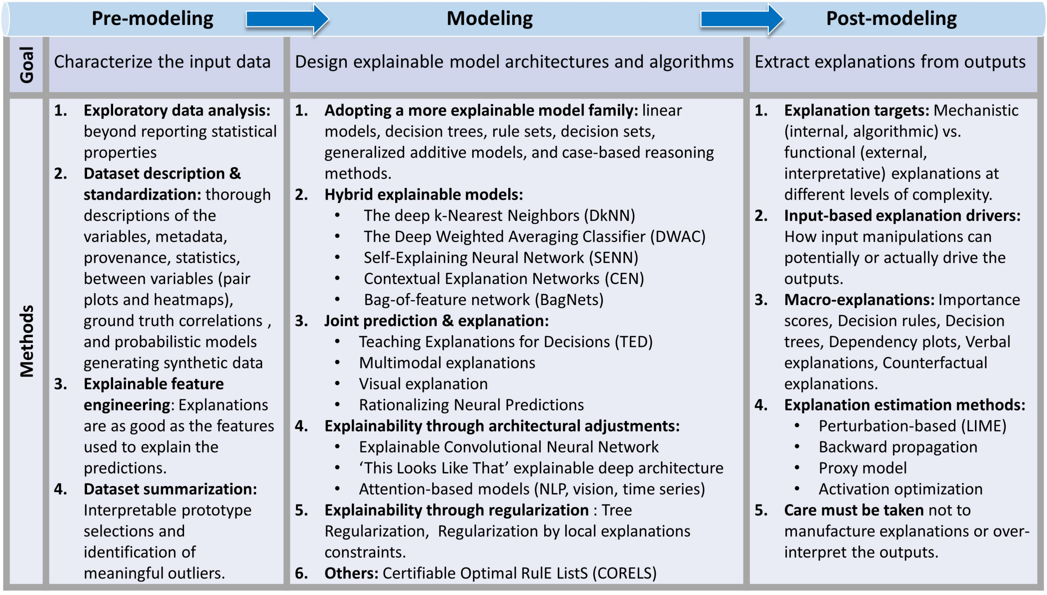


Fig. 2.15 Consolidated XAI Research in 2020
from [127]

More recently, a focus on model and data explainability can be seen in the XAI survey literature [124]. This work aims to provide an in-depth comparative analysis of XAI methods to help better inform various stakeholders who have some involvement with XAI-enabled application development. As noted in the work, their survey has focused on "…approaches to develop XAI applications, covering tools, and technologies for XAI and related concepts to aid implementation in AI-based systems." The taxonomy proposed in that paper can be seen in Figure 2.17. Here, we can see the shift in thinking over the last few years, with the classification of works now being placed into an "Explanation Pipeline" structure. This structure for categorising work defines the author's perceived major steps when dealing with the generation of explanations from AI systems in its entirety, from initial source data to the explanations and assessment thereof. The concept of an explanation pipeline has been introduced in several works, such as its use to structure the XAI developments in neuroscience [128], it itself an adaptation of [129].

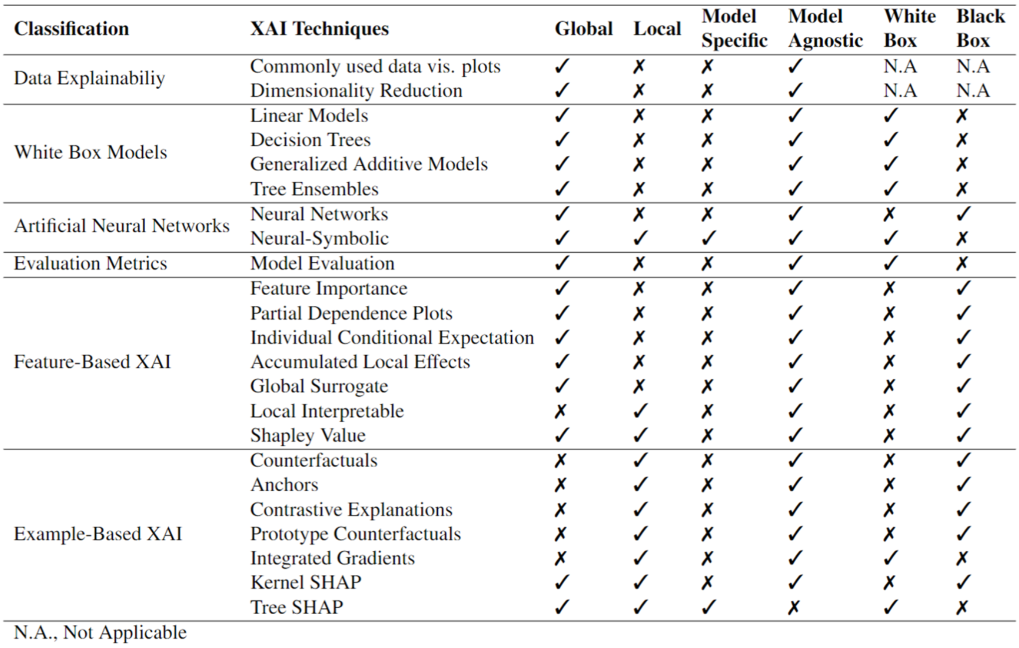
An example of such an explanation pipeline can be seen in Figure 2.16. Here, we see the concept showing three major areas of explanation generation - Pre-Modelling, Modelling and Post-Modelling. They are broadly in line with the earlier attempts at taxonomic categorisation



	Pre-modeling	Modeling	Post-modeling
Goal	Characterize the input data	Design explainable model architectures and algorithms	Extract explanations from outputs
Methods	1. **Exploratory data analysis:** beyond reporting statistical properties 2. **Dataset description & standardization:** thorough descriptions of the variables, metadata, provenance, statistics, between variables (pair plots and heatmaps), ground truth correlations , and probabilistic models generating synthetic data 3. **Explainable feature engineering**: Explanations are as good as the features used to explain the predictions. 4. **Dataset summarization:** Interpretable prototype selections and identification of meaningful outliers.	1. **Adopting a more explainable model family:** linear models, decision trees, rule sets, decision sets, generalized additive models, and case-based reasoning methods. 2. **Hybrid explainable models:** • The deep k-Nearest Neighbors (DkNN) • The Deep Weighted Averaging Classifier (DWAC) • Self-Explaining Neural Network (SENN) • Contextual Explanation Networks (CEN) • Bag-of-feature network (BagNets) 3. **Joint prediction & explanation:** • Teaching Explanations for Decisions (TED) • Multimodal explanations • Visual explanation • Rationalizing Neural Predictions 4. **Explainability through architectural adjustments:** • Explainable Convolutional Neural Network • 'This Looks Like That' explainable deep architecture • Attention-based models (NLP, vision, time series) 5. **Explainability through regularization** : Tree Regularization, Regularization by local explanations constraints. 6. **Others:** Certifiable Optimal RulE ListS (CORELS)	1. **Explanation targets:** Mechanistic (internal, algorithmic) vs. functional (external, interpretative) explanations at different levels of complexity. 2. **Input-based explanation drivers:** How input manipulations can potentially or actually drive the outputs. 3. **Macro-explanations:** Importance scores, Decision rules, Decision trees, Dependency plots, Verbal explanations, Counterfactual explanations. 4. **Explanation estimation methods:** • Perturbation-based (LIME) • Backward propagation • Proxy model • Activation optimization 5. **Care must be taken** not to manufacture explanations or over-interpret the outputs.

Fig. 2.16 Early Explanation Pipeline Definitions [128]

of XAI work shown in [127], with the addition of the Pre-modelling step which covers what modern taxonomies would now call data explainability.



Classification	XAI Techniques	Global	Local	Model Specific	Model Agnostic	White Box	Black Box
Data Explainabiliy	Commonly used data vis. plots	✓	✗	✗	✓	N.A	N.A
	Dimensionality Reduction	✓	✗	✗	✓	N.A	N.A
White Box Models	Linear Models	✓	✗	✗	✓	✓	✗
	Decision Trees	✓	✗	✗	✓	✓	✗
	Generalized Additive Models	✓	✗	✗	✓	✓	✗
	Tree Ensembles	✓	✗	✗	✓	✓	✗
Artificial Neural Networks	Neural Networks	✓	✗	✗	✓	✗	✓
	Neural-Symbolic	✓	✓	✓	✓	✓	✗
Evaluation Metrics	Model Evaluation	✓	✗	✗	✓	✓	✗
Feature-Based XAI	Feature Importance	✓	✗	✗	✓	✗	✓
	Partial Dependence Plots	✓	✗	✗	✓	✗	✓
	Individual Conditional Expectation	✓	✗	✗	✓	✗	✓
	Accumulated Local Effects	✓	✗	✗	✓	✗	✓
	Global Surrogate	✓	✗	✗	✓	✗	✓
	Local Interpretable	✗	✓	✗	✓	✗	✓
	Shapley Value	✓	✓	✗	✓	✗	✓
Example-Based XAI	Counterfactuals	✗	✓	✗	✓	✗	✓
	Anchors	✗	✓	✗	✓	✗	✓
	Contrastive Explanations	✗	✓	✗	✓	✗	✓
	Prototype Counterfactuals	✗	✓	✗	✓	✗	✓
	Integrated Gradients	✗	✓	✗	✓	✓	✗
	Kernel SHAP	✓	✓	✗	✓	✗	✓
	Tree SHAP	✓	✓	✓	✗	✓	✗

N.A., Not Applicable

Fig. 2.17 XAI Techniques Vs. Taxonomy [124]

When contrasting the taxonomies shown in Figure 2.15 and Figure 2.17, we see that the work in [124] has categorised many of the research routes and methods, choosing to focus on the specific stages of explanation generation such as the source data, the model itself, features extracted from the models and the example-based system that aim to extract explanations from the solutions. Across all of these, they have also shown whether each approach can generate global or local explanations and whether they are model agnostic. XAI taxonomies are dynamic and continue to be refined over time. Notably, the work in [124] as shown in Figure 2.17 classifies linear models in "White Box Models" as primarily Global explanations, whereas other perspectives acknowledge their ability to generate local explanations through techniques like LIME which is discussed later in this chapter. This highlights the ongoing development of XAI frameworks and the evolving understanding of explanation types and their generation.

Evolutionary Computing for XAI (EC-XAI)

The term "explainability" is rarely found in the EC literature as stated in [130], however, there are many works that broach the topic in some manner including the "innovization" concept to help identify commonalities between solutions found on the pareto-front [131, 132]. The paper outlines the significant potential in the application of traditional XAI techniques to EC optimisation methods such as those outlined in this literature review. They note that "...despite the differences in the problem formulation (ML vs optimisation), using or adapting XAI techniques to explain the processes used within EC to tackle search problems will improve the accessibility of such methods to a wider audience, increasing their uptake and impact."

This work also points out a set of questions which has arisen from the application of EC and ML techniques in the area, the most related to this work being "Why has the algorithm obtained solutions in the way that it has?".

These questions also broadly overlap with those proposed in the "ART" principals for AI [100] and also align well with the key terms in XAI - Understandability, Comprehensibility, Interpretability, Explainability and Transparency. For the remainder of this thesis, we will use the term "EC-XAI" to refer to the growing subset of XAI research which predominately aims to address the rarity of explainability work when directly applied to EC optimisation methods. The taxonomy shown in Figure 2.17 has been chosen to base our categorisation of the many possible methods of XAI explanation generation, including the work contained within the EC-focused subset of EC-XAI. Table 2.5 shows what we consider to be the main aspects of these classifications that are directly applicable to EC-XAI research. For each of the categories, the table summarises the main aspects discovered during this literature review.

XAI Aspect	Domain
Data Explainability	Data used in AI models, Data Sources, and Preprocessing.
Model Explainability	Explaining internal workings of AI models and identify the contribution of individual features.
Interpretable Model Design	Focuses on designing AI models that are inherently interpretable.
Example-Based Techniques	Uses specific instances to illustrate how an AI model makes decisions.

Table 2.5 Overview of Different Aspects of Explainable Artificial Intelligence (XAI)

2.3.3 Data Explainability

The term Data Explainabiliy is used to define methods to provide an end user with some level of insight into the datasets being used. These methods are model agnostic, being used on both the source data and some outputs of ML and EC models. They also tend to generate explanations on a global level. These methods include exploratory analysis, the visualisation of the data and dimensional reduction techniques to visualise the relationship between variables and present the user with a lower dimension and more interpretable dataset.

Examples of visualisation techniques include using Principal Component Analysis [133] and t-Distributed Stochastic Neighbourhood Embedding [134]. These aim to reduce the dimension of the original datasets and visualise the resulting low-dimension sub-spaces created in order to help identify latent structure and variable correlations that may not be obvious from an initial investigation of the data.

Table 2.6 shows a subset of the most commonly used dimension reduction techniques for the purpose of data analysis and visualisation that would be categorised as being used for Data Explainability by our taxonomy. While not exhaustive, this selection highlights the wide range of possible methods for this purpose and their common usages.

PCA and MCA

Principal Component Analysis (PCA) involves the calculation of linear combinations of the variables in each population such that the resulting principal components are orthogonal to each other. This process “produces linear combinations of the original variables to generate the axes, also known as principal components or PCs” [135] and is a method of reducing the dimensionality of a given problem by reducing the dimensions in the solution to a subset of PCs. Each component is calculated such that it captures maximum variance while ensuring

Method	Common Usages	Data Types
Principal Component Analysis	Data visualization, noise reduction, feature extraction.	Continuous numerical data.
Multiple Correspondence Analysis	Analysis of survey data, marketing research.	Categorical data.
t-Distributed Stochastic Neighbourhood Embedding	High-dimensional data visualization, image processing.	High-dimensional data
Linear Discriminant Analysis	Classification problems, supervised learning, face recognition.	Labeled datasets
Uniform Manifold Approximation and Projection	Data exploration, complex data visualization.	High-dimensional data / Mixed data.

Table 2.6 Comparison of Mathematical Decomposition Methods

orthogonality to the other components. An example of these components can be seen in Figures 2.18 and 2.19.

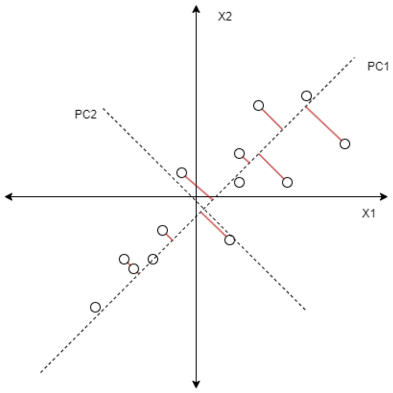


Fig. 2.18 PCA Component Directions

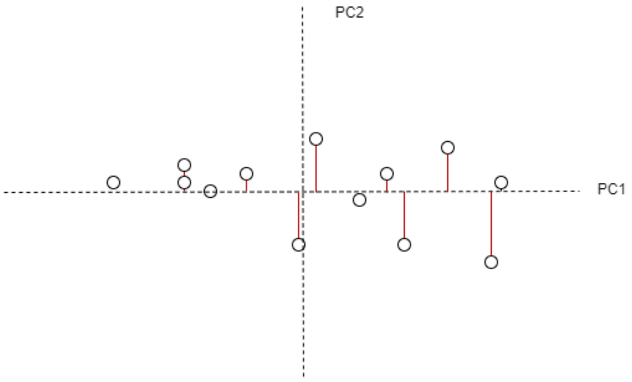


Fig. 2.19 PCA Rotated Subspace

$$
\begin{aligned}
\text{SVD:}\quad & \mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T \\
\text{PCA:}\quad & \widetilde{\mathbf{X}} = \mathbf{X}\mathbf{V} \\
\text{MCA:}\quad & \mathbf{F} = \mathbf{G}\mathbf{H}^T
\end{aligned}
\tag{2.20}
$$

In PCA as shown in Equation (2.20), $\mathbf{X}$ represents the original, mean-centred data, while $\mathbf{V}$ is the matrix of eigenvectors. These eigenvectors can be calculated via singular value decomposition (SVD) of the covariance matrix of $\mathbf{X}$, and $\widetilde{\mathbf{X}}$ is the transformed dataset, or the new, projected dataset in the principal component space. The principal components PC_k and their coefficients (eigenvectors) up to PC_k can be represented as follows in Equation (2.21)

$$\begin{aligned} PC_1 &= v_{11}X_1 + v_{12}X_2 + \ldots + v_{1m}X_m \\ PC_2 &= v_{21}X_1 + v_{22}X_2 + \ldots + v_{2m}X_m \\ &\vdots \\ PC_k &= v_{k1}X_1 + v_{k2}X_2 + \ldots + v_{km}X_m \end{aligned} \tag{2.21}$$

Here, v_{ij} corresponds to the element in the i-th row and j-th column of the matrix $\mathbf{V}$, indicating the weighting of the j-th original variable (X_j) in the i-th principal component (PC_i). These weights come from the eigenvector matrix $\mathbf{V}$, and they serve to rotate the original data space to the new principal component space. The components are ordered such that PC_1 explains the most variance within $\mathbf{X}$ and each subsequent PC_i explains less than the previous.

In MCA, $\mathbf{G}$ is the indicator matrix, a one-hot encoding of the original variable categories, derived from the categorical data, while $\mathbf{H}$ represents a matrix of factor scores, which are calculated using SVD on $\mathbf{G}$. The matrix $\mathbf{F}$ is the transformed dataset in the multiple correspondence analysis space, representing the rotated dataset where the rotation is achieved by the factor scores in $\mathbf{H}$. This allows us to link PCA to MCA such that they are both capable of projecting the search trajectories into rotated subspaces in which either variance or variability in observed variable categories is maximised. Chapter 4 contains a more in-depth breakdown of the process of creating the Complete Disjunctive Table which is used as the indicator matrix in MCA.

There are many advantages to using either PCA for continuous data or MCA for categorical data for dimension reduction and visualisation. In PCA, the process is a well-understood, deterministic process that preserves the variance structure of the original data. Usually when performing PCA, a small number of the initial components are needed to represent a significant proportion of the variance in the data and the process is considered non-destructive. The main benefit of MCA is its ability to be used to gain a similar level of analysis that PCA provided but on categorical, nominal data. This is especially useful for visualising any relationships between specific categories found in the data and can reveal underlying patterns or clusters. The main disadvantage to PCA is its presumption of only linear relationships between variables in the dataset. Any non-linear relationship will, at best, not be detected by the correlation matrix used to calculate the components and at worst, skew the resulting components. For both approaches, the interpretability of the results can be dependent on the dimensionality of the data. With very high dimensional data, the resulting components can be difficult to interpret and the relationships they highlight may not be intuitive.

T-SNE

t-Distributed Stochastic Neighbour Embedding (t-SNE) is a student t-distribution adaptation of the Stochastic Neighbour Embedding algorithm [136] and has been shown to excel a generating visualisations of very high-dimensional datasets. This process involves the use of the Kullback-Leibler Entropic Divergence metric – the measure of entropy or information loss between two distributions – to produce a lower-dimensional projection of the original dataset. The distributions used are generated from the observed Euclidean distances between points and are converted to join probabilities.

As noted in [137] "...The similarity of datapoint x_j to datapoint x_i is the conditional probability, $p_{j|i}$, that x_i would pick x_j as its neighbour if neighbours were picked in proportion to their probability density under a Gaussian centred at x_i." The main stages of the t-SNE process are shown in Figure 2.20 and outlined in Equations 2.22 to 2.24.

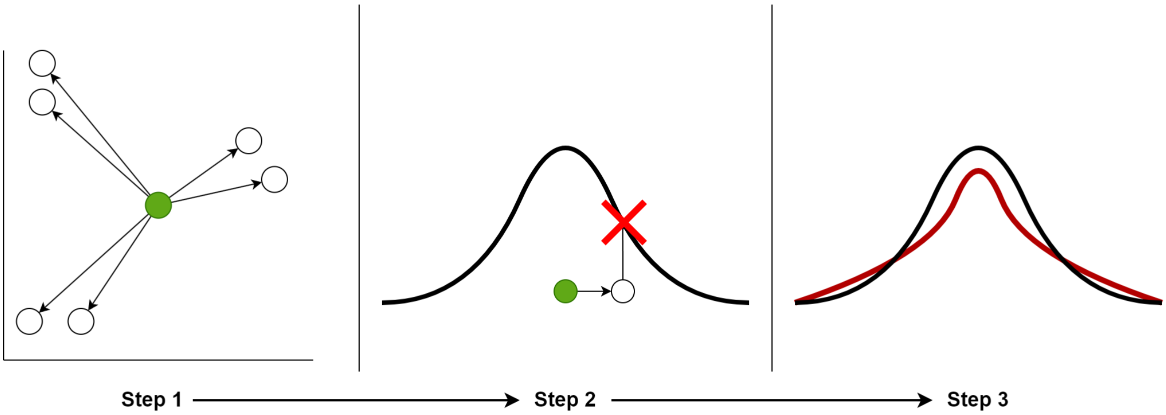


Fig. 2.20 t-SNE Example

The figure shows each of the major steps of this process. Firstly, for each point of interest, the Euclidean distance is measured between it and all other points. This is then plotted against a Normal Distribution curve as seen in Step 2. The distance between the points and the curve is then measured. This is done to find the unscaled similarity scores between each point and the point of interest. Step 3 shows how, in t-SNE, a "Student's t-distribution" is used. The difference between the normal distribution and the t-distribution is shown in step 3. The t-distribution is used to calculate the distribution of points in the subspace and is then compared to the results from the original dataset. The aim in t-SNE is to minimise the cost function such that "If the map points y_i and y_j correctly model the similarity between high dimensional data points x_i and x_j, the joint probabilities $p_{ji} = q_{ji}$. Therefore, t-SNE aims to find a low dimensional representation that minimizes the mismatch between p_{ij} and q_{ji}." [138]. The Equation (2.22) to (2.24) show how this is calculated.

$$P_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)} \tag{2.22}$$

$$Q_{j|i} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||y_i - y_k||^2)^{-1}} \tag{2.23}$$

$$C = \sum_i KL(P_i|Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}} \tag{2.24}$$

Here, $P_{j|i}$ shows the probability that given points x_i and x_j in the original, high-dimension space are in close proximity to each other. $Q_{j|i}$ is the probability that points y_i and y_j, the projections of these points into the lower dimension subspace, are also in close proximity. θ_i is the variance of the Gaussian distribution that is centred on point x_i. The cost function C is minimised using a gradient descent method. Here, P_i is the conditional probability distribution over all data given the point x_i, Q_i is the conditional probability distribution over all other subspace points, given point y_i as described in [139]. This cost function highlights t-SNE's ability to retain local structure in the original data when projected to the subspace as there is a large cost associated with assigning highly separated points as neighbours in the subspace and a much lower cost for points already in close proximity in the original space. This is in contrast to PCA and MCA, in which global variance is maintained across the projection.

T-SNE offers several advantages over other dimension reduction techniques as its ability to capture non-linear variable relationships, something PCA is unable to do. It is also a very effective cluster identification process when dealing with high-dimension data. The disadvantages of this approach include, depending on the user's requirement, its inability to preserve global structure when compared to PCA. When clusters are identified, their relative position in the original space, which may be of importance to the user, may not be preserved. Another disadvantage is its non-deterministic nature. The stochastic elements such as the randomly selected starting points and use of gradient descent lead it to be, as its name suggested, an overall stochastic approach. Given the same set of inputs, the output cannot be guaranteed to be the same.

LDA

Linear Discriminant Analysis (LDA) is a method of class feature identification that can be used a dimension reduction technique by generating a set of latent variables, much like PCA, representing the relationships in the data as linear combinations that maximise variance. LDA however maximises the variance between observed classes in the data to the within-class variance in the dataset. This process achieved maximum class separability in the lower dimension subspace as demonstrated in Figures 2.21 and 2.22.

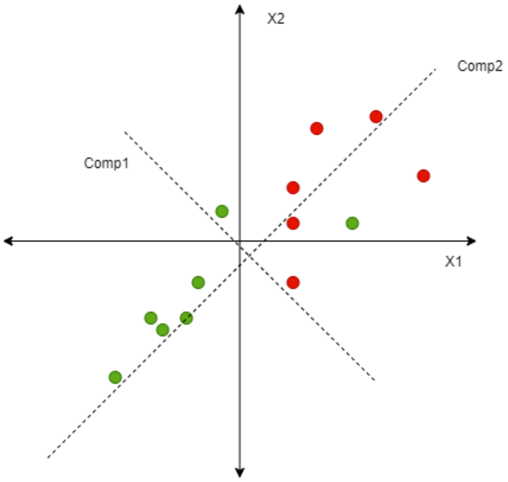


Fig. 2.21 LDA Components.

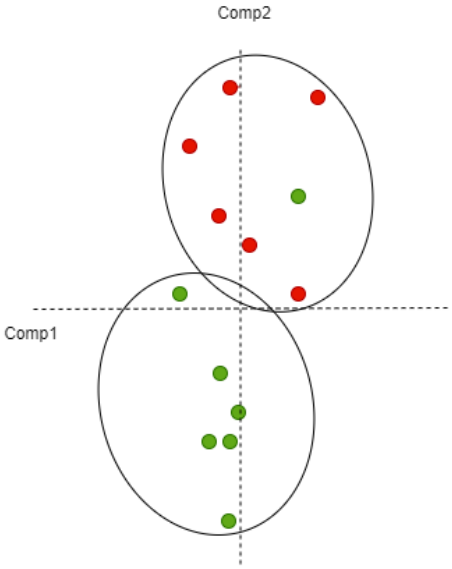


Fig. 2.22 LDA Project to Max. Class Distance.

$$\begin{aligned}
&\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x}\in D_i} \mathbf{x} \\
&S_W = \sum_{i=1}^{c} S_i \text{ where } S_i = \sum_{\mathbf{x}\in D_i} (\mathbf{x}-\mathbf{m}_i)(\mathbf{x}-\mathbf{m}_i)^T \\
&S_B = \sum_{i=1}^{c} n_i(\mathbf{m}_i-\mathbf{m})(\mathbf{m}_i-\mathbf{m})^T \\
&S_W^{-1} S_B \mathbf{v} = \lambda \mathbf{v} \\
&\mathbf{y} = W^T \mathbf{x}
\end{aligned} \tag{2.25}$$

Shown in Equation (2.25), $\mathbf{m}$ is the mean vector across all classes i in the dataset, $\mathbf{m}_i$ is the mean vectors for each class i in the dataset and $\mathbf{S}_B$ is the between-class scatter matrix that is calculated for all observations between each class of the labelled data. The number of occurrences of the class i is contained in n_i. $\mathbf{S}_W$ is the within-class scatter matrix of all observations within the same class in the labelled data and $\mathbf{S}_B$ is the between-class scatter matrix. Once these have been calculated, the eigenvalue problem must be solved. This is accomplished as shown in the equation where $\mathbf{v}$ are the eigenvectors and λ are the eigenvalues. To project the decomposed data into a lower dimension space by selecting the top k eigenvectors based on the largest eigenvalues, to create a transformation matrix W. The projection is then achieved where x is the original data and y is the data after being projected into the lower dimension space. A major advantage that LDA can provide is its ability to improve classification performance by utilising the maximal class separation it detects. LDA however shares the same possible disadvantage as PCA in that it assumes a linear relationship between features and classes. Additionally, a known issue with LDA is its inability to accurately deal with the small sample size problem [140], however as seen in [141] work has been done to alleviate this.

UMAP

Uniform Manifold Approximation and Projection (UMAP), much like t-SNE, is a dimension reduction technique that uses gradient descent in its determination of the lower dimension representation of data. While t-SNE minimised the KLd between the joint probabilities of the high and low dimension projections, UMAP uses gradient descent to minimise the differences between a graph embedding of both datasets, maintaining both local and global structure in its projection. This is accomplished by approximating the manifold on which the original data lies using fuzzy sets.

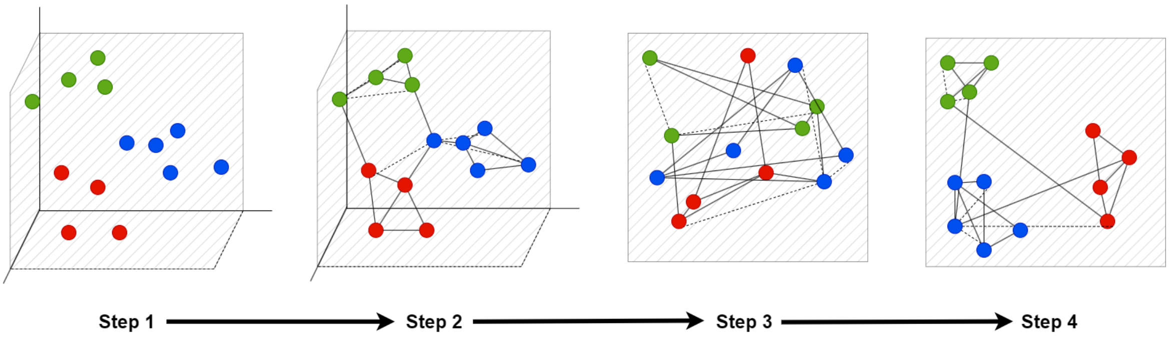


Fig. 2.23 Uniform Manifold Approximation and Projection

This process is shown in Figure 2.23. Here, step 1 is the data in its original form. Step 2 shows the weighted graph representation of the data that is calculated using the fuzzy sets. These weights are represented as in Equation (2.26) This is then used to show the data in a lower dimensional space as seen in step 3. Finally, using stochastic gradient descent, the cost function as shown in Equation (2.27) is minimised to find a representation in the lower dimension space that minimises the cross-entropy loss between the two graph representations. In this equation, $d(y_i, y_j)$ is the distance between the two points y_i and y_j in the lower dimension space. This is done to find an optimal low-dimensional representation, preserving as much of the structure as possible.

$$w_{ij} = \text{weight for edge } (i, j) \text{ in the weighted k-neighbour graph} \tag{2.26}$$

$$C = \sum_{i,j} w_{ij} \log \frac{w_{ij}}{e^{-d(y_i,y_j)}} + (1 - w_{ij}) \log \frac{1 - w_{ij}}{1 - e^{-d(y_i,y_j)}} \tag{2.27}$$

As noted, this process shares many similarities with t-SNE including the use of stochastic gradient descent to find the optimal projection.

Clustering

Clustering methods such as DBScan [142] can be used to extract a visual explanation regarding structures found in the input data. These explanations can help identify data quality issues, biases, and pre-processing requirements. Shown in Figure 2.24 are the main steps in the DBScan method for data cluster visualisation.

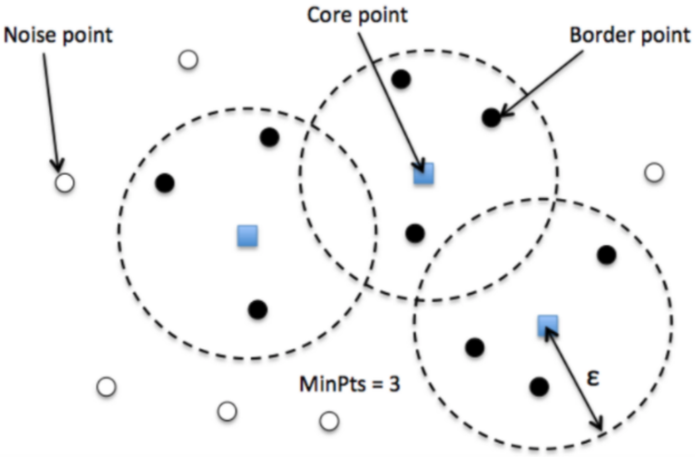


Fig. 2.24 Density-Based Scan Clustering

Here, all points in the dataset are labelled as unvisited. In the figure, for each unvisited point in the dataset, the Core is assigned to any point that has at least *MinPts* neighbours within a given distance ε. The Border point is any point that has at least one Core point within ε distance of itself. A Noise point is any point that does not fall under the Core and Border point classification. To achieve the clustering, the algorithm follows these steps:

1. Each unvisited point in the dataset is selected and it is determined whether it is a Core point. If not, a new point is selected.
2. When a Core point is found it becomes the start of a new cluster. All points within ε distance are considered reachable and added to this cluster.
3. Each unvisited point within the new cluster is now inspected using these same rules.
   (a) If the point is within ε of a Core point and meets the $MinPts$ requirements it is also considered a Core point. All of its neighbours are added to the cluster.
   (b) If the unvisited point is within ε of a Core point but does not have enough $MinPts$ neighbours, it is considered a Border point.
4. This process is repeated until no more points can be added to the cluster. A new unvisited point in the dataset is selected and the process begins again. This is repeated until all points lie within a cluster or are labelled as Noise.

2.3.4 Model Explainability

Linear Models

Linear models and logistic regression are fundamental techniques in statistics and machine learning, widely used for their simplicity, interpretability, and effectiveness in various applications. They are particularly relevant in the context of Explainable Artificial Intelligence (XAI) due to their transparent and understandable nature. Linear models are used to predict a response variable in the form of a linear combination of predictor variables, typically regression tasks, where the goal is to predict a continuous outcome. Generally, a linear regression model takes the form shown in Equation (2.28).

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n + \varepsilon \tag{2.28}$$

Here, the response variable is Y and the predictor variables are $X_1, X_2, \ldots, X_n$. Each variable in the linear combination has an associated coefficient, shown in the equation as $\beta_0, \beta 1, \ldots, \beta_n$. Each linear equation also has an error term, ε, which is used to represent the uncertainty in the model. The model operates under the assumption that these errors conform to a Gaussian distribution, indicating that they encompass variations in both negative and positive directions. An example of linear regression for a single predictor variable is shown in Figure 2.25. Here, as there is only one predictor variable, the resulting model can be represented as a single line. In this figure, we show also what the results of using 2 predictor variables look like.

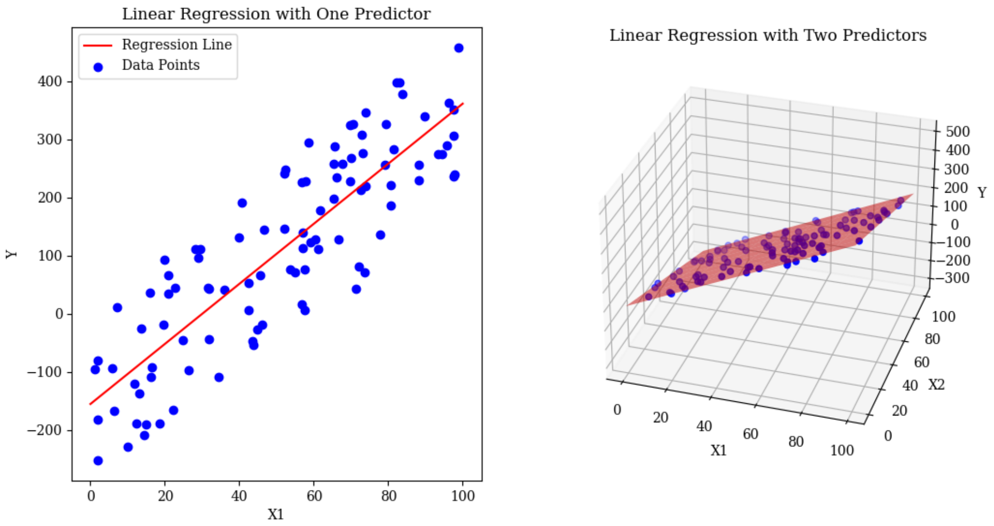


Fig. 2.25 Linear Regression Line and Plane

The results of the application of linear regression on a randomised dataset are shown. The left image shows the results of the regression when only one predictor variable is used. The resulting model can be expressed in the form of a line equation with values $Y = 5.17 * X_1 + (-155.46)$ and in the 2nd plot, showing 2 predictor variables, the equation results in a hyperplane with a function value of $Y = 4.96 * X_1 + (-3.00) * X_2 + 1.15$. This can be generalised to the form:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

Here, for n predictor variables, we can calculate the hyperplane in n-dimensional spaces. In this equation, Y is the dependant variable, $X_1, X_2, \ldots, X_n$ are the predictor variables, and $\beta_0, \beta_1, \ldots, \beta_n$ are the coefficients or weightings. The constant β_0 represents the intercept point on the Y-axis. As the output of the model is the weighted sum of the variables, the relative importance, or impact, of each feature is represented in the magnitude of its respective coefficient, giving an insight into the importance of each feature in the prediction.

Logistic Regression

It is possible to extend linear regression to classification problems. This can be achieved by using a logistic function to push the ML models predictions to a value between 0 and 1. The logistic Equation is shown in (2.29).

$$P(y^{(i)} = 1) = \frac{1}{1+\exp(-(\beta_0+\beta_1 x_1^{(i)},\ldots,\beta_p x_p^{(i)}))} \tag{2.29}$$

Here, $y^{(i)}$ is the binary result of a two-class classification, where the classes are 0 or 1. $P(y^{(i)} = 1)$ is the probability that $y^{(i)}$ will take the value of 1. The remainder of the equation is designed to convert the linear combination of variables and coefficients found in linear regression into a probability value between 0 and 1. This utilises a sigmoid function, shown in Equation (2.29), to map the linear combination to the probability. Figure 2.26 shows the S-shaped function curve.

As shown in [116], it is possible to move from the linear regression method shown to the logistic regression required for classification. The right-hand side of the linear equation can be altered to include the logistic function, as shown in Equation (2.29). This results in the values output from the model to take a value of either 0 or 1, as shown in Figure 2.27

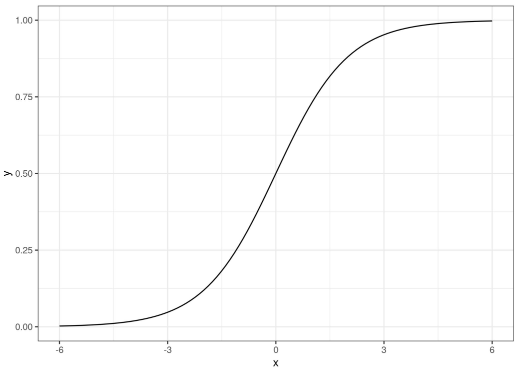


Fig. 2.26 Logistic Function

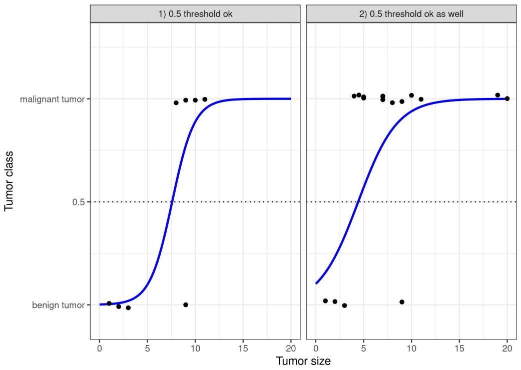


Fig. 2.27 Logistic Regression Class Threshold

Here, the model aims to find the decision boundary between two possible cancer diagnoses based on tumour size. Many of the advantages of linear regression are also applicable to logistic regression as well as the advantage that, due to the use of probabilities in its function, it is possible to provide the probability of an observation taking a specific class which can be very useful. The disadvantages of this process however are that, much like linear regression, variable interactions may need to be added manually as a separate feature and logistic regression suffers from the issue of "Complete separation". This issue occurs when, in the dataset, there is a feature that could perfectly separate classes. This is because "...the weight for that feature would not converge because the optimal weight would be infinite.". However, at an implementation level, this issue may be solved via weight penalisation or the defining of a probability distribution of weights before model training.

PDPs and ICEs

The Partial Dependence Plot (PDP) is used to measure and display the marginal effect of varying a small subset, usually one or two, features, on the prediction made by a machine learning model [143]. By varying these features, it is possible to gain insight into the nature of the relationship between the target predictor and the features altered, showing whether it is linear, monotonic or other depending on the model being examined. An example of this would be when the process is applied to a linear regression model, the resulting PDP will reflect a linear relationship.

The PDP depends on the partial dependence function, which for a linear regression, is defined as shown in Equation (2.30).

$$\hat{f}_s(x_s) = E_{X_C}[\hat{f}(x_s, X_C)] = \int \hat{f}(x_s, X_C) dP(X_C) \tag{2.30}$$

$$\hat{f}_s(x_s) = \frac{1}{n}\sum_{i=1}^{n} \hat{f}(s_s, s_C^{(i)}) \tag{2.31}$$

Here, x_s are the set (usually 2) of features to be used in the creation of the PDP. X_C are the remaining features that the ML model, $\hat{f}$ has been trained on. The combination of both x_C and x_s generates the total feature space x. The process works by marginalizing "...the machine learning model output over the distribution of the features in set C, so that the function shows the relationship between the features in set S we are interested in and the predicted outcome." [116]. This creates a function that will depend only on the features in the set S. The partial function is then estimated. This is done by using the average values in the training data, known as using the "Monte Carlo" method of estimation as shown in Equation (2.31). As the process uses all of the instances in the dataset, it can be said that PDPs are a form of Global explanation as it is detecting the global relationship of the selected features and the outcome of the ML model.

Figure 2.28 is an extract from [116]. This figure shows a PDP based on the interaction of the age and number of pregnancies a patient has against their predicted chances of developing cancer. The plot shows that, at the age of 45, there is a clear increase in the chance of a cancer diagnosis. For patients with 1 or 2 pregnancies who are also under the age of 25, there is a lower band of probability that cancer will develop when compared to those with more than 2 or no pregnancies at all. It should be noted however, as in the study, that there is always a chance that the PDP is representing a correlation between the features that may not be causal.

When dealing with large feature sets and visualisation, it is often preferable to focus on a small number of features. Similar to the PDP, it is possible to create an Individual Conditional

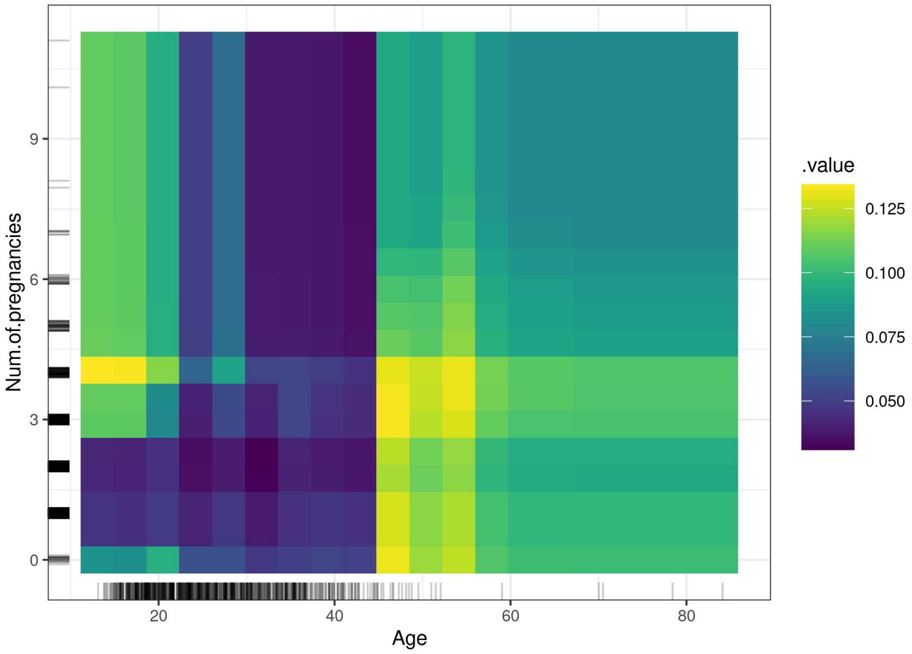


Fig. 2.28 Partial Dependancy Plot

Expectation plot (ICE). These are calculated in a similar fashion to the PDP however there is one significant difference. When creating ICEs, they focus on the relationship between a single feature and the model's prediction, with each observation being used individually rather than the average value as with the PDP. This allows each observation in the data, and its effect on the prediction, to be shown. An example set of ICEs is shown in Figure 2.29.

In this example, taken from [144], the figure shows a set of ICE plots representing the independently calculated ICE results of both temperature and humidity on the number of bikes rented in the OpenML bike rental benchmark dataset [145]. Each line represents a single observation and its partial dependence value. shown in the dashed orange line is the average, or PDP results.

As noted in [144], There is an advantage to using ICEs over PDPs which is that the ICE is capable of detecting heterogeneous relationships. PDP however can detect non-linear relationships. Advantages to using either for explainability include their use in detecting anomalies in the dataset and overall offering a clear view of the relationship between a small subset of features and the outcome of an ML model. There are drawbacks to both approaches however. In PDPs, as the average value is used it can hide heterogeneous effects and both approaches suffer from the issue of high dimensionality in the data - plotting a large number of features can become less interpretable.

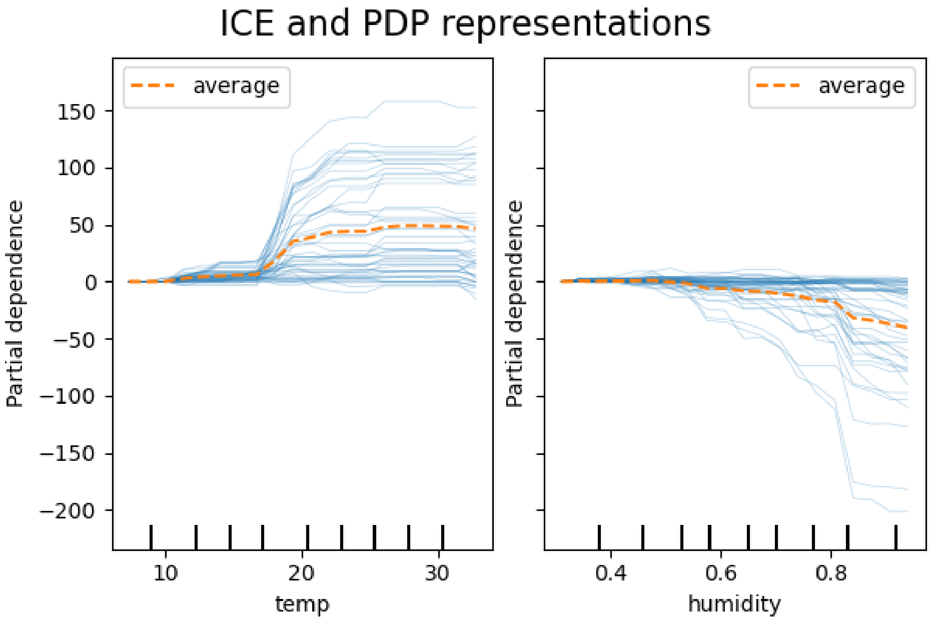


Fig. 2.29 Individual Conditional Expectation Plots from [144]

Landscape Analysis

Landscape analysis and Exploratory Landscape Analysis (ELA) involve characterizing different properties of optimisation problems. Initially a theoretical idea introduced in 1932 [146] in evolutionary computation, it has expanded into a practical tool for a broad spectrum of optimisation and machine learning applications. This growth reflects the field's increasing relevance to complex problem-solving and algorithm behaviour understanding. This can be seen in its growing presence in conferences including EvoStar and the Genetic and Evolutionary Computation Conference (GECCO).

The work in [147] provides a historical perspective on the evolution of ideas within this domain. This survey identified 22 examples of techniques, which have served as foundational concepts and have continued to evolve and adapt over time. More recently, the work of [148] aims to re-evaluate the research area, highlighting the progress in research since their first survey and identifying several key areas where landscapes have been. These techniques have been used to shed light on the behaviour of algorithms, a critical aspect in various domains. In [149], landscape analysis is used to better our understanding of algorithm search behaviour in dynamic environments while [150] applies landscape analysis to local search algorithms, aiding in the understanding of their strengths and limitations. This has also been applied to multi-objective evolutionary algorithms [151, 152].

Further methods that can be used to better our understanding of algorithm search involve the creation of Local Optima Networks (LONs) [154] and Search Trajectory Networks (STNs)

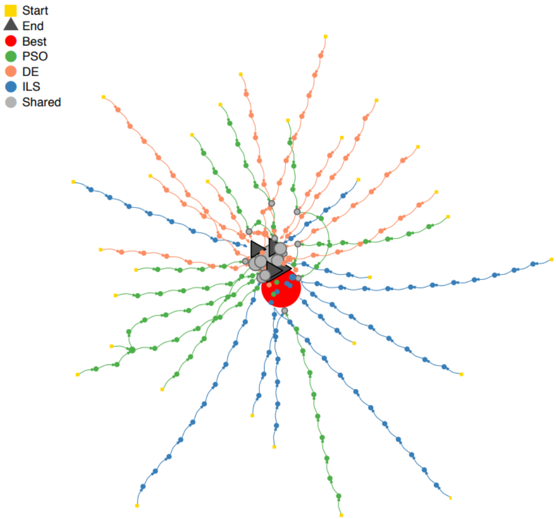


Fig. 2.30 A STN Showing Algorithm Search on the Solomon Benchmark [153]

[155, 153]. Local Optima Network visualisations can be used to illustrate an algorithm's capacity to transition between local optima or basins of attraction, enabling the comparison of algorithmic search behaviours by simplifying the landscape into a connected network of visited local optima. LONs have been used to extend the visualisation techniques commonly used in single-objective optimisation to the multi-objective domain using a Network of Pareto local optimal solutions (PLOS-Net) [156].

STNs are instrumental in highlighting algorithm behavioural differences as they navigate problem landscapes. This can be seen in Figure 2.30 which shows an STN of a collection of optimisation algorithms solving the Solomon bench-marking problem. Unlike LONS, the nodes in STNs do not need to specifically represent local optima. Instead, these nodes can represent solutions such as the best within a generation. The network edges map these consecutive solutions, or locations in the search, mapping commonly used pathways by algorithms. This network visualization offers insights into the landscape's structure and optimisation algorithm navigation.

Recently the field has witnessed developments in "Explainable Landscape-Aware optimisation" and analysis [157, 158]. These advancements represent a shift towards predicting algorithm performance based on landscape analysis, signifying a growing interest in understanding and harnessing the insights gained from studying problem landscapes.

2.3.5 Interpretable Model Design

LIME

Local Interpretable Model-Agnostic Explanations (LIME), introduced in [159], has increased in popularity in recent years as a model agnostic method of generating explanations for individual predictions of the underlying model. This is achieved by creating a more interpretable model that approximates the predictions of the based model it has been created from. To generate the explanations, individual solutions are perturbed and the initial base model is used to predict the fitness of these new observations. This new dataset is then used to train the more simplistic, local surrogate model. In this way, LIME creates a local explanation in that it aims to explain an individual observation rather than the whole dataset.

$$e(x) = \arg\min_{g \in G} \mathbf{L}(f, g, \pi_x) + \Omega(g \tag{2.32}$$

The more interpretable model used in LIME can take many forms, such as a linear regressor or decision tree as outlined earlier. The explanations generated by the system, as shown in Equation (2.32), are used to generate solutions after training has occurred. Here, they aim to minimise the loss **L**, a loss measure such as the mean squared error, which is used to measure how close the explanation is to the prediction of the more complex model. To help with interpretability, the system aims to use as few features as possible in the data set, known as model complexity ($\omega(g)$), where G is the set of possible explanations generated, such as the set of all possible linear regression models. g is an element of g, such that best approximates the fitness value in the neighbourhood of x. In [116] they state that the measure of π_x is a proximity or similarity measure such as cosine similarity or euclidean distance that is used to define the size of the neighbourhood around the instance x, however the exact metric used is domain specific. It is used to assign a higher weight to explanations (g) that are closer or more similar to x. In the equation, f represents the prediction of the initial model of solution x. They outline the main steps of LIME as:

1. Select your instance of interest for which you want to have an explanation of its black box prediction.

2. Perturb your dataset and get the black box predictions for these new points.

3. Weight the new samples according to their proximity to the instance of interest.

4. Train a weighted, interpretable model on the dataset with the variations.

5. Explain the prediction by interpreting the local model.

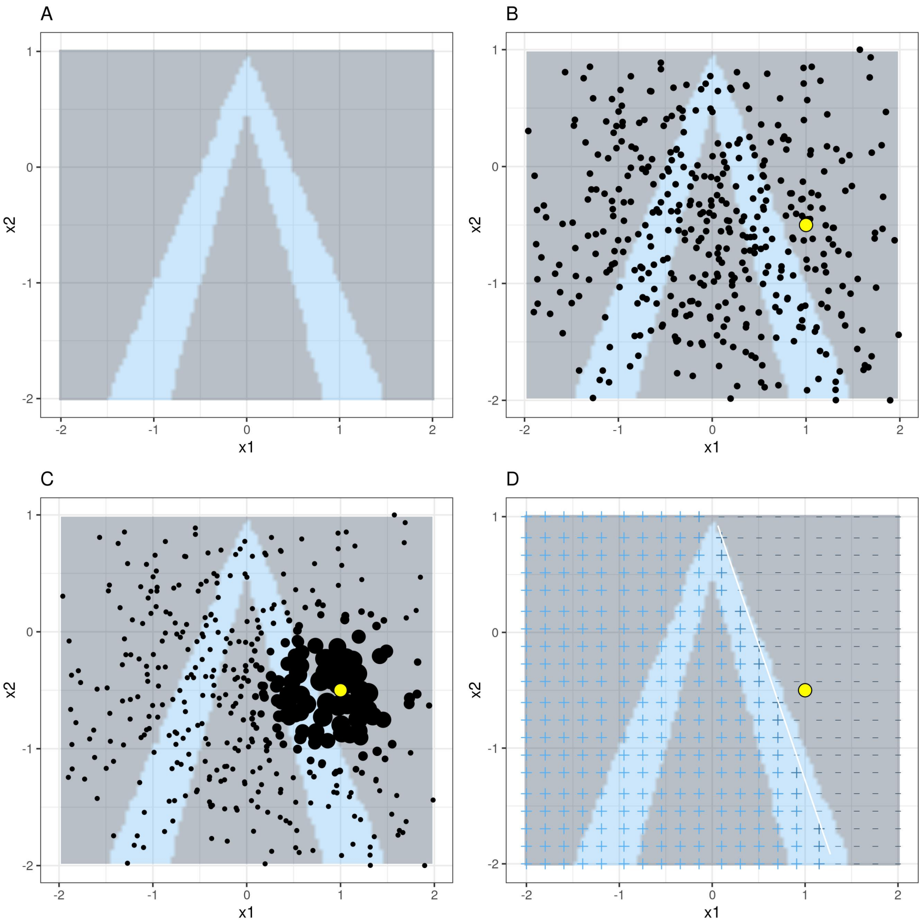


Fig. 2.31 LIME Fitting [116]

This process can be seen in the images in Figure 2.31 as an example of LIME being used on tabular data. Figure 2.31.A shows the search space that the initial model will be working in. In 2.31.**B**, the original model's solution we are interested in is the yellow point. Using this, a perturbed new set of solutions is generated as in step 2. They are then weighted according to their proximity to the initial point of interest as in step 5. The results of this are shown in Figure 2.31.**C** - the closer the new solution is to the point of interest, the higher its weighting. The grid in Figure 2.31.**D** is marked with indicators representing the classifications made by the locally trained model based on weighted samples. Additionally, a white line indicates the decision boundary where the probability of class=1 (dark area of the plot) is 0.5.

A key advantage of this method is that it is model agnostic. This allows the LIME approach to be performed on the predictions of any ML model which is of great benefit when the underlying model is black-box in nature. This feature has led to a rise in popularity in the XAI community.

LIME has the benefit that, due to it creating a more interpretable surrogate model, it is considered model-agnostic. This allows the method to be applied to the predictions of any ML model. This is particularly valuable if the underlying model is not inherently interpretable, such as common 'black box' models as deep neural networks or ensemble methods. Additionally, the local explanations generated by LIME are more accessible to non-experts, facilitating better understanding and trust in machine learning models. LIME does have some disadvantages, such as the explanations provided by LIME are inherently local as they are based on a set of perturbed solutions from one initial point of interest. This means that the findings may not accurately represent the model's behaviour in other areas of the search space.

Shapley

In machine learning, the game-theory term shapley value refers to a value that is used to fairly distribute the contribution of each feature in a problem to the prediction given by a specified ML algorithm [118]. This approach provides a method of understanding the contribution of each feature to a specific prediction much like LIME, however the underlying principles are different.

The contribution of each feature in the problem, i, can be calculated as shown in Equation (2.33).

$$\phi_i(f,x) = \sum_{S\subseteq\{1,\ldots,n\}\setminus\{i\}} \frac{|S|!\cdot(n-|S|-1)!}{n!}\left(f_x(S\cup\{i\})-f_x(S)\right) \tag{2.33}$$

Here, $\phi_i(f,x)$ is the Shapley value for the feature i that has been used by the ML model f, and x is a solution instance. The set of all features is shown as $\{1,\ldots,n\}$, with S being a subset of the features that does not contain i. The size of the subset S is denoted as $|S|$, with a size of $n-1$, with n being the number of all features. The prediction of the ML model is shown in the equation as $f_x(S)$ in which, as mentioned, S is the set of all features excluding i. Lastly shown as $f_x(S\cup\{i\})$ is the prediction made by the model using features in both S and i. This process involves the repeated retraining of the model using different subsets of the features in order to determine the average marginal contribution based on all possible combinations of features. The combinatorial nature of these calculations leads to the factorials seen in Equation (2.33). Figure 2.32 shows the results of calculating the Shapley values for all features in the data set used in [160].

In this figure, the impact of a set of variables on predicting house prices is shown. They state that "...SHAP values of all the input features will always sum up to the difference between baseline (expected) model output and the current model output for the prediction

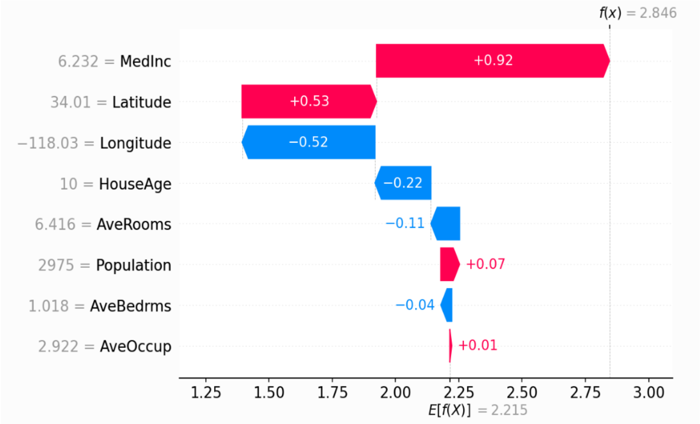


Fig. 2.32 Shapley Value Example

being explained.". Because of this, the results in Figure 2.32 are generated by adding features one at a time until the predictions made match that of the original model, providing the Shapely values for each feature. As noted in [160], when considering the output of the Shapley values, It should be noted that to calculate the total Shapley value, the sum of differences between the source model's predictions and the target prediction across all features is used. This generates the waterfall plot in 2.32. This shows the positive and negative impact on the prediction. In this example, it is shown that a higher median income in the block group drives the predicted house price up, while, to a lesser extent, the older the house the lower the predicted value. This process can be extended to look at all observations in a dataset and typically uses the mean, absolute Shapley value across all observations for each feature.

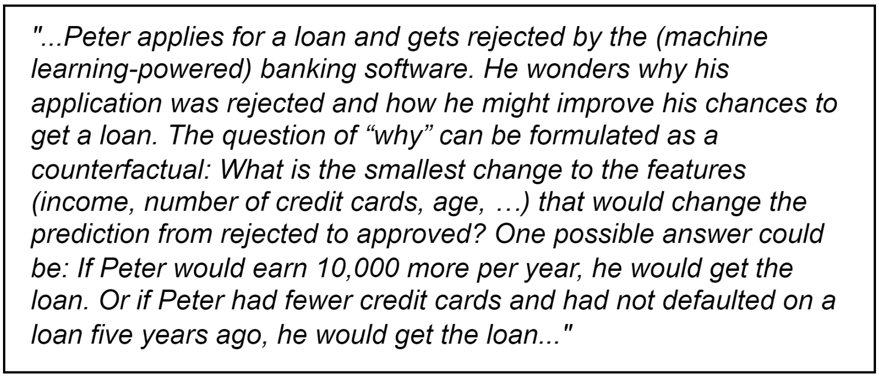
2.3.6 Example-Based Techniques

Counterfactuals

In explainable ML, counterfactuals can be considered a human-friendly, interpretable form of explanation regarding the potential change needed to achieve a target value or goal in optimisation. As an example, a counterfactual could take the general form of:

- "If Variable X has taken value Y, The output would have been Z".

This form of explanation can easily be seen as having great potential to end users in explaining key decisions made by a system, by allowing them to essentially query the ML model on the minimum required change to achieve the targeted outcome. A commonly used example for introducing counterfactuals is that of a bank loan application outcome. When a customer applies for a bank loan, it is possible that they are rejected. As in [116], their example reads:



"...Peter applies for a loan and gets rejected by the (machine learning-powered) banking software. He wonders why his application was rejected and how he might improve his chances to get a loan. The question of "why" can be formulated as a counterfactual: What is the smallest change to the features (income, number of credit cards, age, …) that would change the prediction from rejected to approved? One possible answer could be: If Peter would earn 10,000 more per year, he would get the loan. Or if Peter had fewer credit cards and had not defaulted on a loan five years ago, he would get the loan..."

Fig. 2.33 Counterfactual Example from [116]

The criteria for counterfactuals, first formalised in Tetlock. et. al [161] and extended in J. S. Levy [162], when put into the context of explainable AI and ML can be read as:

- **Clarity**: In ML counterfactuals, the hypothetical changes to the input data or model parameters should be clearly defined, ensuring that the alternate outcomes or predictions are logically derived and easily interpretable.
- **Minimal-Rewrite Rule**: The counterfactual scenario in ML should involve minimal alterations to the input data or model configuration, focusing on key changes that significantly impact the outcome, while keeping the rest of the data or model structure largely unchanged.

- **Cotenability**: All elements of the ML counterfactual scenario should be mutually compatible, ensuring that changes in data points or model parameters do not introduce inconsistencies or contradictions within the model's framework.

- **Consistency with Well-Established Theoretical Generalizations**: Counterfactuals in ML should adhere to established principles and theories in the field, such as statistical learning theory, to ensure that the hypothetical scenarios are grounded in scientifically valid concepts.

- **Historical Accuracy**: In the context of ML, this criterion would translate to data accuracy, where the counterfactuals should be based on accurate and realistic data representations, deviating from the original dataset only in specific, intentional ways.

- **Temporal Proximity**: For ML counterfactuals, this would involve focusing on changes that have immediate or short-term effects on the model's output, avoiding long-term or far-reaching speculations that are less certain and harder to quantify or interpret.

To generate the counterfactuals, Wachter et al [163] developed a method involving the minimisation of a loss function that aims to reduce the distance between two specific metrics - the distance between the predicted outcome and the counterfactual results, and then the absolute differences of feature values between instance x and the counterfactual. This approach, however, is not able to deal with categorical variables.

The work of Dandl. et. al [164] has condensed these criteria to four specific requirements that can be formalised and used as part of an optimisation process for the creation and refinement of counterfactuals. This approach is capable of dealing with categorical variables, using a smaller set of criteria:

- **o_1 A counterfactual instance produces the predefined prediction as closely as possible** – A significant element of this criterion is that it may require the adjustment of decision boundaries to ensure that the required prediction is possible.

- **o_2 A counterfactual should be as similar as possible to the instance regarding feature values**

- **o_3 It should produce multiple diverse counterfactual explanations**

- **o_4 A counterfactual instance should have feature values that are likely**

These four new criteria, $\mathbf{o_1}$ - $\mathbf{o_4}$, are used as part of the objective function in their approach as seen in Equation (2.34) - (2.39)

$$L(x, x', y', X^{obs}) = \big(o_1(\hat{f}(x'), y'), o_2(x, x'), o_3(x, x'), o_4(x', X^{obs})\big) \tag{2.34}$$

$$o_1(\hat{f}(x'), y') = \begin{cases} 0 & \text{if } \hat{f}(x') \in y' \\ \inf\limits_{y' \in y'} |\hat{f}(x') - y'| & \text{else} \end{cases} \tag{2.35}$$

$$o_2(x, x') = \frac{1}{p} \sum_{j=1}^{p} \delta_G(x_j, x'_j) \tag{2.36}$$

$$o_3(x, x') = ||x - x'||_0 = \sum_{j=1}^{p} \mathbb{I}_{x'_j \neq x_j}. \tag{2.37}$$

$$o_4(x', \mathbf{X}^{obs}) = \frac{1}{p} \sum_{j=1}^{p} \delta_G(x'_j, x_j^{[1]}) \tag{2.38}$$

$$\delta_G(x_j, x'_j) = \begin{cases} \frac{1}{\widehat{R}_j} |x_j - x'_j| & \text{if } x_j \text{ numerical} \\ \mathbb{I}_{x_j \neq x'_j} & \text{if } x_j \text{ categorical} \end{cases} \tag{2.39}$$

Here, o_1 outlines how the prediction of the counterfactual value should be in close proximity to the required output, by minimising the Manhattan metric between $\hat{f}(x')$ and y'. o_2 specifies how the counterfactual needs to be similar to the initial solution x, measuring the difference between the two using Gower distance [165]. The third objective, o_3, uses the L_o norm to count the number of features that have been changed to maintain that distance and create a sparse set of feature updates. The final objective, o_4, uses the training data to ensure that the created instances have feature values that are "likely" to be seen in the dataset, again by measuring the Gower distance between the new instance created and the nearest point in the observed, training data. In the equations, p denotes the number of features in the dataset. As mentioned, this method is capable of dealing with both numerical and categorical data types. To achieve this, the Gower distance is shown in Equation (2.39). Here, depending on whether the feature is numerical or categorical, the distance measure is adjusted accordingly, with $\widehat{R}_j$ being the range of feature j.

To optimise this many-objective loss function the NSGA-II algorithm [166] is used. The resulting counterfactuals can then be used to inform the end user of what changes are necessary to reach a desired goal, the objective value of that suggestion against the objectives themselves and the predicted probability of the outcome $\hat{f}(x')$, given the initial result y'.

2.3.7 The Evolutionary Gap

When considering the research taxonomies and extensive collections of work contained in both [127] and [124], many of the methods investigated are geared towards more traditional machine learning. As shown in this literature review, the majority of the techniques highlighted for each main aspect of EC-XAI also fall into this category. A significant feature of the heuristic methods introduced in this chapter is that there is no mathematical theory that extends to all these algorithms - they are stochastic in nature or utilise stochastic mechanisms to generate successive populations of solutions. This often makes such search methods difficult to explain or generate meaningful explanations regarding their decision-making processes. This work in this thesis aims to address this issue by exploiting this shared aspect of successive generation creation. This process generates a search path or trace, representing the EAs gained knowledge of the optimisation problem which may be mined for features found consistently across multiple runs.

Such EAs have been harnessed in EC-XAI to help generate explanations in other systems, such as the generation of neighbourhoods for local, interpretable systems as in local rules-based explanations (LORE) to help explain the decisions made by black-box systems [167]. Additionally, the integration of evolutionary fuzzy systems in XAI, as discussed in [168], emphasizes the significance of EAs in designing interpretable systems.

In EC, the use of classification rules has been growing for several years in the generation of white-box models [169, 170]. Further approaches towards the generation of these more interpretable models in EC include the use of decision trees from which explanations can be extracted [171, 172] and genetic fuzzy systems, to help address the issue of balancing predictive accuracy and interpretability [173]. These are examples of how, in recent years, the EC for XAI research perspective has been growing. This area focuses on how traditional and novel methods of XAI explanation generation can be incorporated or applied directly to EC search methods including population-based metaheuristics.

However, feature-based techniques typically require the generation of a new ML model or surrogate model. Alternatively, methods such as LIME and Shapley involve the creation of a local surrogate or ML model with subsets of the features to help explain specific solutions and the possible drivers towards that decision. This can also be extended to entire datasets. Counterfactual and other example-based techniques aim to explain specific solutions by considering the impact of changing individual variables. As this specific area of XAI research is still relatively new, there exists a gap in the knowledge base regarding population-based metaheuristics. This research aims to help fill the gap of XAI techniques specifically for population-based metaheuristics, and more specifically, those that are aimed at the data-mining of the search trajectories directly.

To highlight this gap in the knowledge base and to give an indication of the level of interest and development in the area, a data-mining exercise was performed. An extensive collection of publications contained in SemanticScholar, a library of works, was created using the methods outlined in [174]. The techniques used in that work involved the search and filtering of papers found in SemanticScholar in which at least 2 of the following terms were found in the title or abstract:

> *xai, (xai), hcxai, explainability, interpretability, explainable ai, explainable artificial intelligence, interpretable ml, interpretable machine learning, interpretable model, feature attribution, feature importance, global explanation, local explanation, local interpretation, global interpretation, model explanation, model interpretation, saliency, counterfactual explanation.*

This body was then merged with a collection of works that was manualy curated by the authors. Citation expansion, in which the top 2000 papers cited in this growing body of works were collected and manually filtered for XAI relevance. This process was then repeated using the same 2-word filtering as the first step. These steps resulted in an initial dataset containing papers up until December 2022 which contained 5199 works from both SemanticScholar and private repositories. To update this dataset to contain papers from 2023 and early 2024, the SemanticScholar database was queried by ourselves using their publicly available API and the resulting 2024 dataset containing an additional 977 papers was merged with the original. The final dataset contained a total of 6176 XAI papers.

To analyse this data, shown in Table 2.7, are some of the main categories of XAI methodologies used in the data-mining exercise. Also shown is a collection of 10 keywords or phrases that typify those categories. For each work contained in the database, the title and abstract (if available) were mined for as many of these terms as possible. This involved the use of the Python package "fuzzywuzzy" [175] to perform fuzzy-matching for each paper's title and abstract combination against the terms in Table 2.7. If a paper was found to have a similarity of 80% or greater to a category then it was assigned that value.

As an example, should a publication title and abstract combination contain a higher total fuzzy-matched similarity score from the search terms of "Neural Networks" than any other category, it was assigned as a neural network paper. This was done to give an indication, as of 2023, of the growth of XAI overall as well as the relatively smaller categories of evolutionary computing focused work in the area.

Also shown in Table 2.7 are the total number of papers from the collection that were determined to belong to each category using the method mentioned.

These results show that there is a significantly larger number of papers in the dataset that have been categorised as "Neural Network" than any other. The evolutionary computing

Category	Search Terms	Count
Neural Networks	neural network, deep learning, convolutional neural network, recurrent neural network, backpropagation, activation function, feedforward neural network, neural architecture search, perceptron, neural network training	2112
Evolutionary Algorithms	genetic algorithm, genetic programming, evolutionary computation, differential evolution, evolution strategy, swarm intelligence, particle swarm optimisation, multi-objective evolutionary algorithm, memetic algorithm, genetic operator	52
Rules-Based Learners	rule-based system, decision tree, rule learning, if-then rules, inductive logic programming, association rule learning, rule induction, decision rules, rule extraction, rule pruning	611
Trajectory & Landscape Analysis	trajectory analysis, landscape analysis, fitness landscape, optimisation landscape, solution trajectory, landscape mapping, trajectory optimisation, landscape characterization, path analysis, landscape metrics	24

Table 2.7 Search Terms for Each Category

based papers in "Evolutionary Algorithms" and work related to our own in "Trajectory and Landscape Analysis" show only 50 and 19 papers respectively. It should be noted however that this is by no means an exhaustive search of all XAI literature available but should give an indication of the level of interest and recent work in both XAI and EC-XAI areas of research. Changing the search terms will alter the results - using more specific terms will more than likely reduce the number of results.

Figure 2.34 shows the steady increase in publications for each year from 2011 to 2024 for each of the categories. The dashed horizontal line indicates the end of 2022 where the dataset only contains the merged Semantic Scholar data and does not contain data from any private repositories. This helps to explain the lower value for 2023 onwards. Overall, it can be seen that over the years, interest in these areas has increased, with the area of our work, "Trajectory & Landscape Analysis" and "Evolutionary Algorithms" showing much lower levels of growth in comparison to the "Neural Network" category, which reflects the overall pattern seen in XAI. We aim to expand on this knowledge base by investigating whether search trajectories can be mined for explanations and explanatory features from the vector-space representations of the search paths taken. It is in this area that the work in this thesis aims to add knowledge, motivated by the apparent gap in EC-XAI focused work.

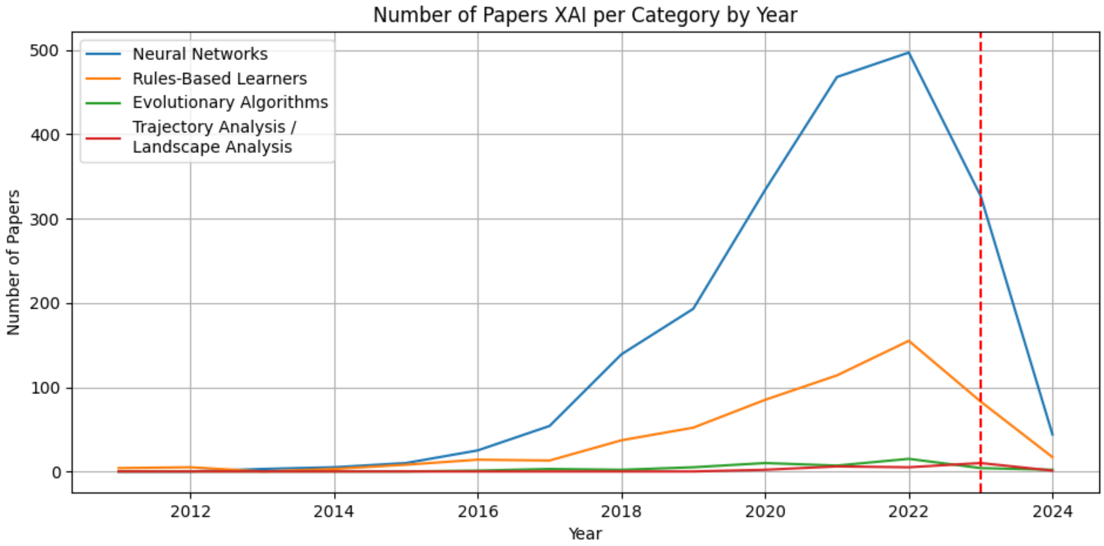


Fig. 2.34 XAI Publication Counts

Chapter 3

Background

3.1 Introduction

In this chapter, we introduce the terminologies and techniques required to formalise our approach to the generation of explanatory features mined from the search trajectories of evolutionary algorithms. This chapter introduces the notation that will be used for the rest of this thesis and how it is used. Throughout this thesis, we examine the trajectories using a variety of different methods designed to extract features from differing problem representations – binary, real-value, and nominal. These string-based and vector-based methods of solution representation have an underlying fitness function which can be expressed as a sum of the basis functions. Because of this, we can derive explanatory features by estimating the parameters of the fitness functions themselves. To achieve this, we perform data mining on the search trajectories for these features from the model of the fitness function. When performing this data mining, we are looking at a situation where an algorithm has made a large delta – moving along the fitness gradient – by moving from a random starting population to one containing a significant proportion of high-quality solutions. A vital component of this work is the use of decomposition - the process of breaking down a complex dataset or problem into simpler, more manageable components that can be used to identify structures. A selection of these methods have been introduced in Chapter 2 however in this chapter, we will discuss the advantages and disadvantages of several approaches, with respect to the requirements of this project. Here, we show an overview of the possible techniques and give a detailed justification for the choice of technique used in the remainder of the thesis, that of Principal Component Analysis. Finally, the selection of problems used is outlined.

3.2 Notation and Problem Encoding

3.2.1 Vector Spaces and Basis Functions

In numerical optimisation, it is important to lay the foundations of how the search space and solutions are represented. This can be achieved by using vector spaces and their associated basis vectors to create a mathematical framework capable of providing a geometric interpretation of the problem at hand. When the optimisation function is presented as such, each point in the search space represents a solution to the problem which lies somewhere on the fitness landscape.

Vector Spaces

In the context of a metaheuristic search, a vector space representation can be used to model the problem where each dimension represents a decision variable and each point in the space represents a possible solution.

As an example, a vector space V over a field $\mathbb{F}$ can be defined by the two vector operators of addition and scalar multiplication. In the vector space V, the following axioms must be true for all $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ in V and any scalars $a, b \in \mathbb{F}$:

- Commutativity of Addition: $\forall \mathbf{u}, \mathbf{v} \in V, \mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$
- Associativity of vector addition: $\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in V, (\mathbf{u} + \mathbf{v}) + \mathbf{w} = \mathbf{u} + (\mathbf{v} + \mathbf{w})$
- Additive identity: $\exists \mathbf{0} \in V, \forall \mathbf{v} \in V, \mathbf{v} + \mathbf{0} = \mathbf{v}$
- Existence of additive inverse: $\forall \mathbf{v} \in V, \exists - \mathbf{v} \in V, \mathbf{v} + (-\mathbf{v}) = \mathbf{0}$
- Closure Under Scalar Multiplication: $\forall c \in \mathbb{F}, \forall \mathbf{v} \in V, c\mathbf{v} \in V$
- Distributivity of vector sums: $\forall c \in \mathbb{F}, \forall \mathbf{u}, \mathbf{v} \in V, c(\mathbf{u} + \mathbf{v}) = c\mathbf{u} + c\mathbf{v}$
- Distributivity of scalar sums: $\forall a, b \in \mathbb{F}, \forall \mathbf{v} \in V, (a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$
- Scalar Multiplication Identity: $\forall \mathbf{v} \in V, 1\mathbf{v} = \mathbf{v}$

If true, a vector $\mathbf{v}$ in a finite vector space V can be represented as:

$$\mathbf{v} = (v_1, v_2, \ldots, v_n) \tag{3.1}$$

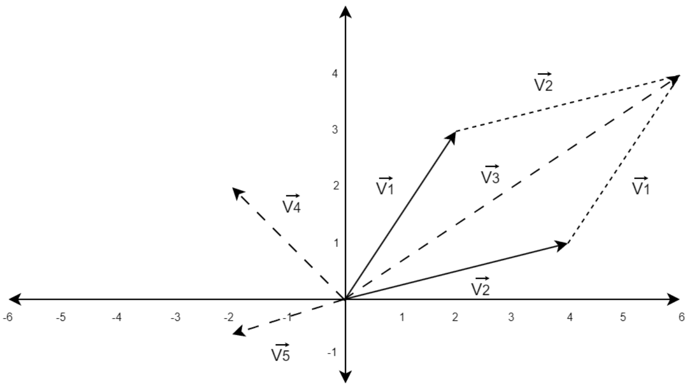


Fig. 3.1 Vectors Example

Here, n is the number of dimensions in the vector space. Consider the vectors $\mathbf{V1} = [2,3]$ and $\mathbf{V2} = [4,1]$. We can demonstrate vector addition, subtraction, and scalar multiplication in the vector space $\mathbb{R}^2$ in Equation (3.5) and in Figure 3.1.

$$V3(Addition) = (V1 + V2) = [6,4] \tag{3.2a}$$

$$V4(Subtraction) = (V1 - V2) = [-2,2] \tag{3.2b}$$

$$V5(Multiplcation) = (-0.5 * V2) = [-2,-0.5] \tag{3.2c}$$

It should be noted that all resulting vectors $V3, V4, V5$ are all still vectors in the $\mathbb{R}^2$ space as per the axioms listed.

Basis Vectors

Basis vectors play a crucial role in representing solutions within vector spaces through optimisation functions. They allow for a representation of vectors as linear combinations, where any vector in the space can be expressed as a sum of these basis vectors, each multiplied by a corresponding coefficient.

An example of a basis is seen in the "standard basis," which is a set of vectors that are both linearly independent and span the entire space - the requirements for any set of vectors to be considered a basis. The standard basis for any space of $\mathbb{R}^n$ consists of a set of n independent vectors, each with a value of 1 in one dimension and 0 in all others. For our $\mathbb{R}^2$ example, the standard basis vectors for this 2-dimensional space are shown below:

$$\text{Let } \vec{e_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ and } \vec{e_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

These vectors, $\vec{e_1}$ and $\vec{e_2}$, effectively form the standard basis for $\mathbb{R}^2$, demonstrating that any vector in this space can be constructed from a linear combination of $\vec{e_1}$ and $\vec{e_2}$. For example, vectors $V1$ and $V2$ in this space can be represented as $V1 = 2\vec{e_1} + 3\vec{e_2}$ and $V2 = 4\vec{e_1} + \vec{e_2}$, respectively.

Function Space Basis

A function space can be represented by a linear combination of a set of basis functions, allowing complex functions to be expressed within a vector space. Any function f(x) can be expressed as the summation of these basis functions, seen in Equation 3.3 such that:

$$f(x) = \sum_{i=1}^{n} c_i \cdot \phi_i(x) \tag{3.3}$$

Here, n is the number of dimensions, c_i are a set of coefficients, and $\phi_i(x)$ are the basis functions. The coefficients in the equation determine the position of the function in the vector space. Consider the function $f(x,y) = 2x + y^2$. This function is polynomial and can be represented in a polynomial function space. As before, this set of basis functions must be linearly independent and must span the whole space. To achieve this, a basis such as that in Table 3.1 could be used. The landscape of this function is shown in Figure 3.2.

$$\begin{bmatrix} \phi_1(x,y) = 1 \\ \phi_2(x,y) = \text{x} \\ \phi_3(x,y) = \text{y} \\ \phi_4(x,y) = \text{x}^2 \\ \phi_5(x,y) = \text{xy} \\ \phi_6(x,y) = \text{y}^2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Table 3.1 Function Basis Example

Here, we see that for each polynomial term, we have a value of 1 for each term and a 0 otherwise. This set of basis vectors spans the whole space and can be used to represent any function value of the variables x and y while following the axioms of a vector space.

This allows us to use the notation in Equation (3.3) to represent the function as a summation of the basis vectors and coefficients. This results in the representation shown in Equations (3.4a) and (3.4b). This allows us to represent the function $f(x,y)$ as a vector in the space defined by our basis functions $\phi(x,y)$ in the form shown in (3.4c).

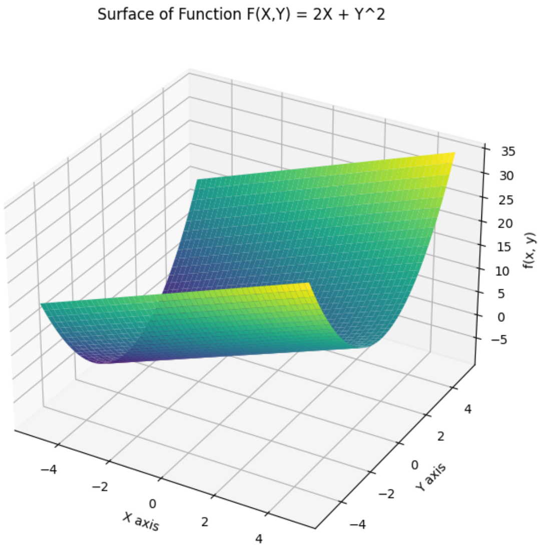


Fig. 3.2 Function Surface Example

$$f(x,y) = c_1 \cdot \phi_1(x,y) + c_2 \cdot \phi_2(x,y) + c_3 \cdot \phi_3(x,y) + c_4 \cdot \phi_4(x,y) + c_5 \cdot \phi_5(x,y) + c_6 \cdot \phi_6(x,y) \tag{3.4a}$$

$$f(x,y) = 0 \cdot 1 + 2 \cdot x + 0 \cdot y + 0 \cdot x^2 + 0 \cdot xy + 1 \cdot y^2 \tag{3.4b}$$

$$f(x,y) = [0,2,0,0,0,1] \tag{3.4c}$$

The basis is capable of representing any polynomial function with up to a second degree in the terms. It is also possible, if the given function is the only function being used, to reduce this basis to the terms for x and y^2; however, only functions in that exact form could be represented. Following this pattern to higher degrees in the matrix can result in very high condition numbers, leading to numerical instability. Here, Strang [40] recommends alternative bases for function spaces that are considerably more efficient, such as those shown in Table 3.2.

Basis	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_i
Fourier	1	$\sin x$	$\cos x$	$\sin 2x$	$\ldots$
Legendre	1	x	$x^2 - \frac{1}{3}$	$x^3 - \frac{3}{5}x$	$\ldots$
Chebyshev	1	x	$2x^2 - 1$	$4x^3 - 3x$	$\ldots$

Table 3.2 Alternative Function Space Basis

3.2.2 Problem Encoding

Encoding is how the data being used in an optimisation problem is represented so that it can be manipulated by the algorithm used. This allows the internal processes to alter, update and score population members according to the fitness function. It is also often necessary to decode the population of solutions at the end of the process as the encoding may not represent the data in real values. This involves converting the solutions from the encoding back to the original form the data was presented to the algorithm. This decoding allows the solution to be presented back to the end user in a form that is more understandable and usable in the context of the problem.

Binary

Binary encoding is the process of converting variables and values of an optimisation problem into a bit string composed of binary digits (bits), where each bit is either 0 or 1 representing a solution in binary space. In EAs, solutions are represented as individuals, each with a set of genes. Each gene has multiple possible alleles, which in this case are 0 or 1. The complete set of alleles in an individual is called its genotype. The process of converting an optimisation problem into a binary problem, taken from Genetic Algorithms and Walsh Functions: Part I [176] can be seen in Equation (3.5a) - (3.5d).

$$f(d) = d^2 \tag{3.5a}$$

$$f(d) = f(g(x)) \tag{3.5b}$$

$$d = g(x) = \sum_{i=1}^{3} x_i 2^{i-1} \tag{3.5c}$$

$$f(x) = \left(\sum_{i=1}^{3} x_i 2^{i-1} \right)^2, x_i \in \{0,1\} \tag{3.5d}$$

This example shows how the problem of maximising $f(d) = d^2$ in Equation (3.5a). Here, as $d = g(x)$ as in Equation (3.5b) the overall problem can be mapped to a binary problem as seen in Equation (3.5c) and (3.5d) where $g(x)$ is the binary representation of d and x_i represents the i-th bit in the binary string. Seen in Table 3.3 is a demonstration of binary encoding for a sample set of solutions.

One drawback of binary encoding is seen when representing real values or values with large ranges. As noted in [177], a problem variable with a range of 1200, when converted to binary, would require 11 alleles to be fully represented. This however would result in

Solution 1	11100101010010101010
Solution 2	10001010010101111001
Solution 3	10001000110101000111
Solution 4	10110111101100100100

Table 3.3 Binary Encoding Example

a total of 848 unused bits as 11 alleles would offer 2048 possible values. This has the potential to generate invalid solutions during operations like mutation and crossover unless such processes are designed to address the issue.

Real-Valued

Real-valued encoding directly represents solutions using numerical values. This approach maintains user interpretability as the values remain unaltered during the encoding process. An example of this can be seen in Table 3.4. Here, each solution is represented by a list of real numbers directly corresponding to the variables in the problem.

Solution 1	3.14, 6.28, 0.07
Solution 2	7.89, 10.01, 0.06
Solution 3	1.81, 4.00, 1.54
Solution 4	4.36, 3.39, 2.77

Table 3.4 Real-Valued Encoding Example

The advantage of real-valued encoding is its efficiency in representing continuous values and maintaining interpretability.

Nominal

Nominal encoding represents solutions using discrete categories or symbols. This approach is useful for problems where solutions belong to a predefined set of options with no inherent order or numerical value. Each category can be represented by a unique string, value or symbol. An example of nominal encoding for a scheduling assignment problem, with each value representing a shift pattern, can be seen in Table 3.5. Here, each solution represents the allocation of shift patterns to a workforce (Morning, Afternoon, Night).

Nominal encoding is efficient for problems with a limited set of discrete categories. However, it might not be suitable for problems where the order of elements is important, in which case permutation encoding could be used. Additionally, mutation and crossover

Solution 1	Morning, Afternoon, Night, Morning, Afternoon
Solution 2	Afternoon, Morning, Night, Morning, Afternoon
Solution 3	Night, Morning, Afternoon, Afternoon, Morning
Solution 4	Morning, Afternoon, Afternoon, Morning, Night

Table 3.5 Nominal Encoding Example for Shift Patterns

operations need to be designed to handle the specific set of categories and ensure valid solutions are generated.

Integer

Integer encoding represents solutions using whole numbers. This approach is useful for problems where solutions involve selecting values from a discrete set with a finite range. It offers a balance between efficiency and interpretability compared to binary encoding for problems with limited value ranges. An example of integer encoding can be seen in Table 3.6. Here, each solution is represented by a list of integers corresponding to the variables in the problem.

Solution 1	27, 31, 34, 57, 61, 62, 88, 93, 96, 98
Solution 2	98, 96, 93, 88, 62, 61, 57, 34, 31, 27
Solution 3	61, 27, 88, 62, 34, 98, 57, 96, 31, 93
Solution 4	93, 62, 61, 57, 98, 27, 34, 88, 96, 31

Table 3.6 Integer Encoding Example

The advantage of integer encoding is its efficiency in representing discrete values within a limited range.

Permutation

Permutation encoding is designed for tasks that require finding the optimal ordering of a set number of tasks. This encoding scheme is commonly used for problems such as the travelling salesperson and flow-shop scheduling, where the goal is to generate a list of locations or tasks in such an order as to minimise the objective function. An example of this can be seen in Table 3.7.

Solution 1	1, 2, 3, 4, 5, 6, 7, 8, 9
Solution 2	1, 2, 3, 6, 7, 8, 9, 4, 5
Solution 3	2, 5, 4, 6, 7, 1, 3, 9, 8
Solution 4	8, 2, 9, 1, 3, 6, 4, 5, 7

Table 3.7 Permutation Encoding Example

3.3 Trajectories

3.3.1 Trajectory Structure

Defining a structure for the populations of solutions generated through optimisation is an important step in this thesis. Each optimisation run generates a series of solutions, structured by generation and each generation represents the algorithm's current position in the search. In this thesis, we define a trajectory T as a collection of EA solution populations X ordered by their generation g as shown in Equation (3.6). This notation allows us to further define an EA search trajectory as a collection of $\mathbb{R}^{gNn}$ solutions ordered by the number of generations, g, the population size N and the problem dimension n.

$$\begin{aligned} T &= [X_1, \ldots, X_g]^\intercal \\ X &= \{x^1, \ldots, x^N\}^\intercal \\ x &= [x_1, \ldots, x_n] \in \mathbb{R}^n \end{aligned} \tag{3.6}$$

Recall in Chapter 2 in which we define the main steps of a population-based metaheuristic as the Evaluation, Learning and Update steps. The formalisation of a search trajectory structure used in this thesis is dependent on the fact that all population-based metaheuristics share two common traits concerning these steps. Firstly, all must utilise the fitness function of a problem to determine the quality of a solution as in the Evaluation step. Secondly, all must generate some form of positional change at the end of the Update step, regardless of any internal operators. The final output of the Evaluation step is a positional update representing the incremental change in the algorithm's understanding of the optimisation problem. By basing our trajectory on this trace of the algorithm's search, our analysis of the trajectories remains a post-hoc, model-agnostic approach to search trajectory analysis.

3.3.2 Fitness Quartiles

In the analysis of optimisation search trajectories, it is sometimes necessary to group sections of those trajectories. This can be due to computation limitations or for interpretability reasons

- reducing the overall number of results shown, at the cost of accuracy, to be more readily understood. To achieve this, we use a method that groups the search trajectories into four quartiles. One method is to group the data based on the generation number. As an example, should there be a total of 100 generations in an optimisation run, then each quartile will contain all data for each group of 25 generations. Another approach is to split the search trajectory into four groups of interest. Here, the methodology for determining the generation number at which a specific fitness threshold is reached within an optimisation run is shown, indicating the groups of interest. The fitness threshold is used as a measure for understanding the progress of an optimisation process. This allows for the splitting of the run into four distinct stages of interest based on the overall change in solution fitness achieved. The process of calculating the generation number at which this occurs can be seen in Equations (3.7) to (3.9).

$$f_g^* = \min_{x \in X_g} F(x_g) \tag{3.7}$$

$$\Delta f^* = f_0^* - f_l^* \tag{3.8}$$

$$g_q = \min\{g : f_0^* - f_g^* \geq q\Delta f^*\} \tag{3.9}$$

These equations show how the generation number at which a certain fitness threshold can be calculated in a given optimisation run. Here, $F(X_g)$ is the collection of fitness values of all solutions at generation g. f_g^* represents the fitness of the best-found solution in a given generation. For these calculations, f_0^* represents the initial best-found solution in the starting population of an optimisation run. Δf^* is the total reduction in fitness from the initial best solution to the final solution, here shown as f_l^*. The fitness thresholds are represented by q - a value of 0.33 would be used to calculate the generation at which approximately 33% of all fitness change has been observed. Typical values for this calculation would be those for 25%, 50% and 75%. This generation number is then averaged across all runs to determine the mean generation at which a threshold is achieved. This is then used to partition the optimisation runs into quartiles for later analysis.

3.3.3 Decomposition Techniques

As noted in the XAI section of the literature review, decomposition techniques themselves can be categorised as a method of explanation generation in some form. There are a significant number of techniques that can be used for the decomposition of complex data into smaller, more manageable components and each has its advantages and disadvantages in comparison to the other. In 2009, many of these possible methods were reviewed and a general taxonomy was created to categorise them [178]. In that work, they show that these methods can broadly be broken down into convex and non-convex methods, as shown in Figure 3.3. In that paper, convex and non-convex methods are defined as "Convex techniques optimise an objective function that does not contain any local optima, whereas non-convex techniques optimise objective functions that do contain local optima.". Examples of alternative techniques from that work are Maximum Variance Unfolding (MVU) [179] and Stochastic Neighbourhood Embedding (SNE) [180]

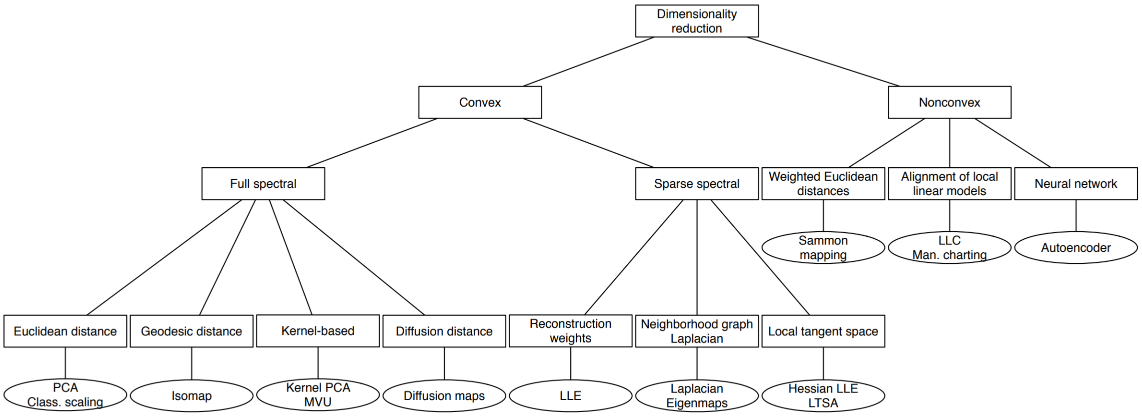


Fig. 3.3 Dimension Reduction Method Taxonomies - Laurens Van Der Maaten, et al, 2009. [178]

Since 2009, there have been developments and new techniques invented, with a prominent example being Uniform Manifold Approximation and Projection (UMAP) [181] in 2018. This method aims to preserve more local neighbourhood structure within the embedding, a known issue with t-SNE. Shown in Tables 3.8 and 3.9 are a selection of the most commonly used methods in machine learning and data science for dimension reduction. The table also outlines their advantages and disadvantages.

Method	Advantages	Disadvantages	Common Usages
PCA (Principal Component Analysis) [182]	Preserves variance. Linear, interpretable link to original features. Efficient computation.	Assumes linearity. Sensitive to scaling. Not for categorical data.	Data visualization, noise reduction, and feature extraction in finance [183], biology [184], and social sciences [185].
MCA (Multiple Correspondence Analysis) [186]	Good for categorical data. Reveals patterns in categories. Relationship with original features.	Less interpretable for continuous variables. Computationally intensive. Linear.	Analysis of marketing research [187] and social science studies [188].
t-SNE (t-Distributed Stochastic Neighbour Embedding) [139]	Reveals clusters in high-dimensional data. Captures non-linear relationships. Good for visualization.	No direct link to original features. Computationally expensive. Sensitive to parameters.	High-dimensional data visualization [139], genomics and bioinformatics [189] and image processing [190].
LDA (Linear Discriminant Analysis) [191]	Supervised; considers class labels. Maximizes class separability. Linear and interpretable.	- Assumes Gaussian distribution. Sensitive to class imbalance. Linear only.	Classification problems, dimensionality reduction [192] in supervised learning, face recognition [193].
UMAP (Uniform Manifold Approximation and Projection) [181]	Captures both global and local structure. Less computationally intensive than t-SNE. Good for visualization.	No direct link to original features. Interpretability can be challenging. Hyperparameter sensitivity.	Complex data visualization [194], genomics [195] and neuroscience [196].

Table 3.8 Comparison of Mathematical Decomposition Methods 1

Method	**Advantages**	**Disadvantages**	**Common Usages**
Autoencoders (Neural Networks) [197]	Captures non-linear relationships. Flexible and powerful. Good for complex data.	- Requires large datasets. Computationally intensive. Black-box nature.	Feature learning [198], anomaly detection [199], image reconstruction [200].
Factor Analysis [201]	Identifies underlying variables. Good for correlated data. Can handle over-determination.	Assumes linear relationships. Requires large samples. Interpretability issues.	Marketing [202] and survey research [203].
Isomap (Isometric Mapping) [204]	Captures geodesic distances. Good for non-linear dimensionality reduction. Reveals manifold structure.	Computationally intensive. Sensitive to noise. No direct link to the original features.	Non-linear dimensionality reduction [205] and manifold learning [206].
NMF (Non-negative Matrix Factorization) [207]	Good for parts-based representation. Non-negativity leads to easy interpretation. Useful for sparse data.	- Requires non-negative data. Not as robust as PCA. Linear.	Text mining [208], image processing [209], and audio analysis [210].
Spectral Clustering [211]	Good for identifying clusters. Works well on non-linear data. Can handle complex structures.	Computationally expensive. Sensitive to choice of similarity metric. No direct link to the original features.	Clustering in image segmentation [212], speech analysis [213] and bioinformatics [214].

Table 3.9 Comparison of Mathematical Decomposition Methods 2

3.3.4 Decomposition Selection

As shown in Tables 3.8 and 3.9, There are other methods of data projection into subspaces that allow for a similar level of analysis as PCA including a modified version of SNE called t-distributed Stochastic Neighbourhood Embedding (t-SNE) [139].

Table 3.10 shows a condensed version of this data to highlight the main benefits of each approach in relation to the main aims of this projects.

Method	**Linear**	**Non-linear**	**Supervised**	**Cat.**	**Viz.**	**Interpretable**	**Link Features**
PCA	X				X	X	X
MCA	X			X	X	X	X
t-SNE		X			X		
LDA	X		X		X	X	X
UMAP		X			X		
Autoencoders		X			X		
Factor Analysis	X					X	X
Isomap		X			X		
NMF	X					X	X
Spectral Clustering		X					

Table 3.10 Features of Decomposition Methods

Here, we show whether the method is capable of dealing with linear and non-linear variable relationships, whether they require supervision, can deal with categorical data ("Cat.") and excel at conveying their findings visually ("Viz."). We also show whether their results are considered interpretable at a glance and if it is possible to link back their findings to the original features or variables of the problem directly. From the table, we can see that no single method works equally well on both linear and non-linear relationships. The main driver in the selection of a suitable method was the ability to link the features directly back to the original variables and features with as little distortion as possible. An added benefit was if the results themselves could be considered interpretable in their own right. With these requirements, PCA and MCA combined (A categorical version of PCA) covered the largest possible number of requirements. Two possible alternatives were LDA and t-SNE.

T-SNE, a popular method of dimension reduction, can similarly be used to project the trajectories into a lower dimension subspace however, a key aspect of t-SNE is its stochastic nature. Each application of this process to the same dataset may result in a different set of values which would be a drawback to our experiments. Noted in [215] regarding this method "...while these methods are an important contribution to dimensionality reduction, they do not produce low-dimensional data as $Y = PX$ for any P".

An additional issue with t-SNE is that, by its nature, it tends to distort any potential key features found from mining the geometrically sensitive relationships between clusters of data points found in the original datasets [137].

T-SNE is excellent at retaining non-linear variance which may be an important further step in our experimentation however, as we mine for features spanning the entire trajectory, a higher importance on global variance rather than local variance was selected to allow the comparison of population members from all stages of the optimisation trajectory. LDA was also considered as a possible alternative, however, as it is a supervised learning method. This reliance on the requirement of class labels was a major drawback. With PCA and MCA, it is possible to apply the methods to all binary, real-valued and nominal problem representations while LDA is not capable of this.

This higher importance being placed on the preservation of geometric structure post-projection and a focus on global variance within the data lent more towards the continued use of PCs for the representation of the trajectories. This preservation of structure and the ability to link back to the original data also allows us to directly relate behaviours in the PCA subspace to the behaviours seen in the input data on a generation-by-generation comparison.

3.3.5 Principal Component Analysis Methodology

The application of PCA to our data results in the creation of a new subspace in which the axes are linear combinations of original variables. The coefficients of these linear combinations are calculated using the Single Value Decomposition (SVD) of the scaled and means-centred input data. This results in a set of latent variables or Principal Components which are directional vectors calculated to maximise variance. The orthonormal matrix generated has orthogonal column vectors which act as the new axis in the subspace, defined in this thesis as P. The directional vectors of this subspace represent a change in basis, in which each function in the new space is comprised of the linear combination of the original variables. This change in basis, resulting in a rotation and scaling of the data, is achieved using a series of linear transformations that preserve the linear relationships and structures present in the original data.

First, the search trajectory $\mathbf{T}$ is decomposed using SVD as shown in Equation (3.10). Also shown in Equation (3.11) is the process of projecting the data into the resulting PCA-derived subspace.

$$\mathbf{T} = \mathbf{U}\Sigma\mathbf{P}^T \tag{3.10}$$

$$\widetilde{\mathbf{T}} = \mathbf{TP} \tag{3.11}$$

The decomposition in Equation (3.10) results in m components, each consisting of n coefficients, which link our original n-dimensional search space to an m-dimensional subspace. This is because each principal component creates a hyperplane in the original coordinate space in which the data points were created. To project the original data into the PC subspaces, the resulting scores can be calculated by the multiplication of the trajectory by these new components, as shown in Equation (3.11). This results in an $n \times m$ matrix representing the original data in a reduced m-dimensional space. The resulting components, **P**, can be defined as in Equation (3.12).

$$\begin{aligned} \mathbf{P} &= \left[\mathbf{p}^1, \ldots, \mathbf{p}^m\right]^T, \quad m \leq n \\ \mathbf{p}^i &= \left[p_1^i, \ldots, p_n^i\right] \end{aligned} \tag{3.12}$$

The approximation of a single solution $\widetilde{x}$ - its projection onto the space spanned by the components - using the first m principal components is shown in Equation (3.13). Here, p_j^i refers to the j-th element of the i-th principal component vector $\mathbf{p}_i$. The term $x \cdot \mathbf{p}_i$ is the dot product of the solution x with the i^{th} principal component vector p_i. This dot product gives the projection of x onto p_i, and multiplying it by p_i reconstructs the contribution of p_i to the projection of x. The summation over m components provides the complete projection, or approximation, of x in the reduced space. By using m components such that $m < n$ we achieve the projection of the solution to the lower-dimensional subspace for use in later analysis.

$$\widetilde{x} = \sum_{i=1}^{m} (x \cdot \mathbf{p}_i)\mathbf{p}_i = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} x_j \cdot p_j^i \right) \mathbf{p}^i \tag{3.13}$$

This allows us to define the complete set of approximated solutions within a given generation of the search trajectory $\widetilde{\mathbf{X}}$, and thereby the search trajectory $\widetilde{\mathbf{T}}$, in the m-dimensional subspace as shown in Equation (3.14). Using this m-dimensional subspace of $\mathbb{R}^n$, we can define a projected trajectory as a collection of $\mathbb{R}^{gNm}$ solutions ordered by g, similarly structured to our original trajectory definition.

$$\begin{aligned} \widetilde{\mathbf{X}} &= \left[\widetilde{x}^1, \ldots, \widetilde{x}^N\right]^{\mathsf{T}} \\ \widetilde{\mathbf{T}} &= \left[\widetilde{\mathbf{X}}_1, \ldots, \widetilde{\mathbf{X}}_g\right]^{\mathsf{T}} \end{aligned} \tag{3.14}$$

3.4 Binary Benchmarking Problems Used

Here, we show a set of binary-string benchmark problems that will be used and discussed in later chapters in relation to problem structure detection and population diversity. Binary benchmark problems are often selected not just for their relative simplicity but because of their known problem structure and optimal solutions. This allows for the comparison of algorithm performance not just in terms of solution quality and run-time analysis, but on their ability to detect and exploit any variable interactions that define the structure of the problem. Noted when discussing problem encoding, Boolean or pseudo-Boolean encoding has also been used to represent real-world problems, such as the evolution of cancer-chemotherapy drug scheduling [216].

3.4.1 The 1D Checkerboard

The 1D Checkerboard function scores the chromosome based on the sum of adjacent variables that do not share the same value [217]. The function is seen here in Equation (3.15).

$$CHECK_{1D}^{l}(x) = \sum_{i=0}^{l-2} \left\{ \begin{matrix} 1, & x_i \neq x_{i+1} \\ 0, & x_i = x_{i+1} \end{matrix} \right\} \tag{3.15}$$

Because the function scores only adjacent variables it is possible to have two possible global maxima. As an example, for a bit string of length 5 the two possible would be [01010] and [10101]. The implementation of the problem used in this thesis also checks the first and last alleles to check if they match. This allows for a total fitness value equal to the bit-string length for an ideal solution.

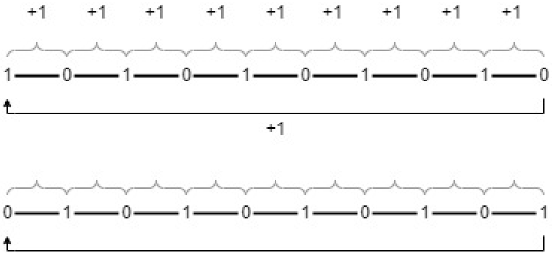


Fig. 3.4 1D-Checker Interactions

Shown in Figure 3.4 is the chain structure and how adjacent alleles with opposite values add to the fitness of a solution in a 10-bit example.

3.4.2 The Royal Road

The Royal Road function scores chromosomes based on collections of variable values based on a specified set of schema that the solution must fulfil in order to score an optimal value

[218]. below, Equation (3.16) specifies the fitness function for the royal road problem with a schema block size of 5, as used in this experiment.

$$R_1(x) = \sum_{i=1}^{5} \delta_i(x) o(s_i), \text{ where } \delta_i(x) = \left\{ \begin{matrix} 1 & \text{if } x \in s_i \\ 0 & \text{otherwise} \end{matrix} \right\} \tag{3.16}$$

As noted in [218] the equation represents the fitness function, such that R_1 is a sum of terms relating to a partially specified schema. The schemata are subsets of solutions that match the partial specification, s_i. As an example, one partially specified schema with a size of 5 could be represented as [11111*****...] where unspecified members are denoted by "*"

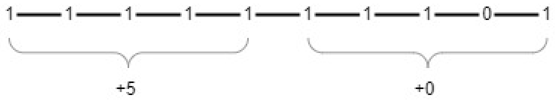


Fig. 3.5 Royal Road Interactions

A given bit-string x is an instance of a specific schema $s, x \in s$ if x matches s in the defined positions within that schema. $o(s_i)$ defines the order of s_i which is the the number of defined bits in s_i. The royal road function was designed to "capture one landscape feature of particular relevance to GAs: the presence of fit low-order building blocks that recombine to produce fitter, higher-order building blocks" [219]. Shown in Figure 3.5 is an example of a 10-bit solution to the problem. Here, we see that when all alleles in the schema have the value of 1, the maximum score of 5 is awarded, however adjacent to this is a schema with only 4 1's. This is scored as 0.

3.4.3 The Trap-5

The Trap-5 concatenated problem is designed to be intentionally deceptive [220]; [221], such that they "deceive evolutionary algorithms into converging on a local optimum. This is particularly a problem for algorithms which do not consider interactions between variables." [66]. As with the Royal Road problem, the bit-strings are partitioned into blocks and their fitness is scored separately. Seen in Equation (5.3a) is the function of a trap of order k.

$$f(x) = \sum_{i=1}^{n/k} trap_k(x_{b_{i+1}} + ... + x_{b_{i+k}}) \tag{3.17a}$$

$$trap_k(u) = \left\{ f_{\text{high}} \text{ if } u = k, f_{\text{low}} - u\frac{f_{\text{low}}}{k-1} \text{ otherwise} \right\} \tag{3.17b}$$

Blocks within the bit-string are scored according to the fitness function in Equation (5.3b). A Trap5 problem with a bit-string length of 10 would have the values n=10, k=5, f_{high} = 5 and f_{low} = 4. The further from the goal of each Trap containing five 1's, the higher the fitness value, with only a maximum achieved when the whole Trap is comprised of 1s, leading the algorithm away from the optimal value, as shown in Figure 3.6.

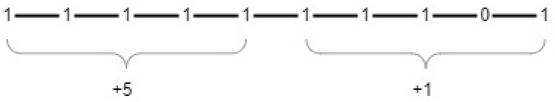
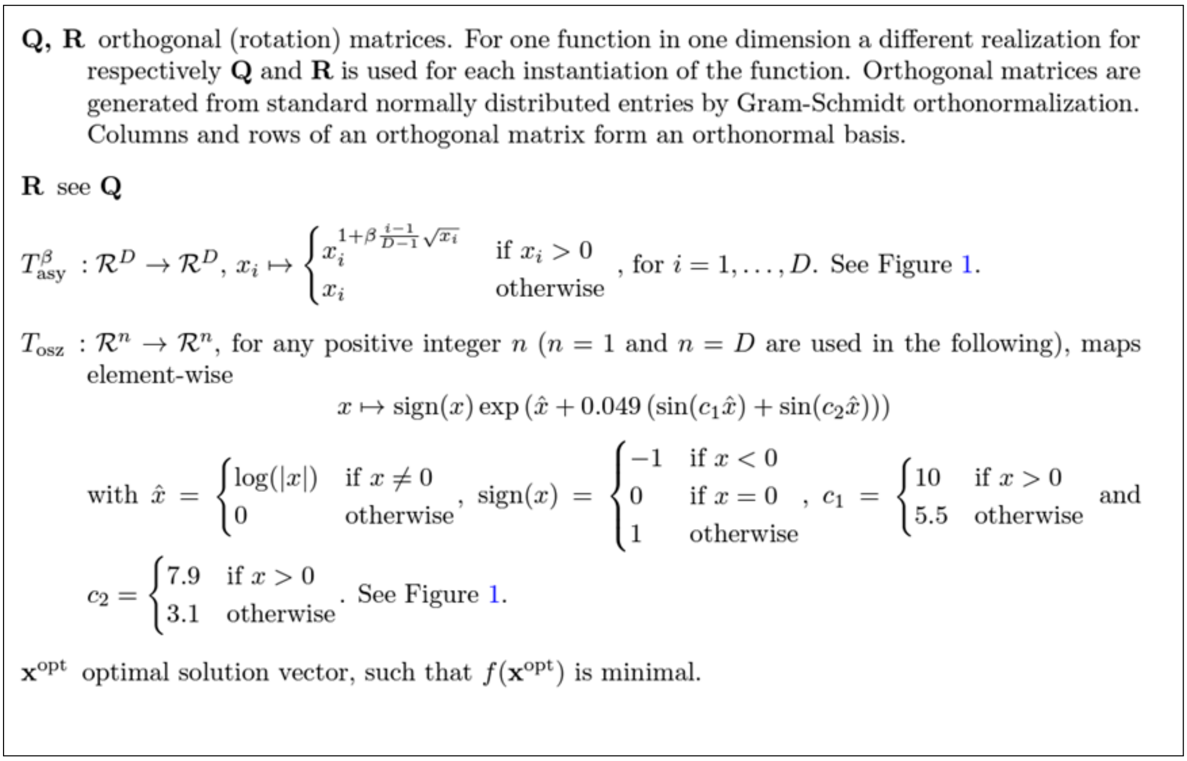


Fig. 3.6 Trap-5 Interactions

3.5 Real-Valued Benchmarks Used

The Black Box Optimisation Benchmarking (BBOB) [222] set of optimisation problems is a well-known and often used problem set for the measurement and comparison of optimisation techniques. Consisting of 24 real-valued parameters, the problem set contains instances of problems with a range of properties including low or moderate conditioning, unimodal, multimodal and both strong and weak global structure examples. It is important to note that while the BBOB problems are named "Black Box", the optima and function values are known for each of the functions beforehand. Shown in Figure 3.7 are the definitions of the terms used in BBOB problem set, taken from [222]. This is followed by a brief outline of four specific instances from the BBOB problem set. These were selected as they represent some of the main attributes of the set - Seperable Functions, Unimodal with High Conditioning and Multi-Modal with either adequate or weak global structure.



Q, R orthogonal (rotation) matrices. For one function in one dimension a different realization for respectively **Q** and **R** is used for each instantiation of the function. Orthogonal matrices are generated from standard normally distributed entries by Gram-Schmidt orthonormalization. Columns and rows of an orthogonal matrix form an orthonormal basis.

R see **Q**

$T^{\beta}_{\text{asy}} : \mathcal{R}^D \to \mathcal{R}^D$, $x_i \mapsto \begin{cases} x_i^{1+\beta\frac{i-1}{D-1}\sqrt{x_i}} & \text{if } x_i > 0 \\ x_i & \text{otherwise} \end{cases}$, for $i = 1, \ldots, D$. See Figure 1.

$T_{\text{osz}} : \mathcal{R}^n \to \mathcal{R}^n$, for any positive integer n ($n = 1$ and $n = D$ are used in the following), maps element-wise

$$x \mapsto \text{sign}(x)\exp\left(\hat{x} + 0.049\left(\sin(c_1\hat{x}) + \sin(c_2\hat{x})\right)\right)$$

with $\hat{x} = \begin{cases} \log(|x|) & \text{if } x \neq 0 \\ 0 & \text{otherwise} \end{cases}$, $\text{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases}$, $c_1 = \begin{cases} 10 & \text{if } x > 0 \\ 5.5 & \text{otherwise} \end{cases}$ and $c_2 = \begin{cases} 7.9 & \text{if } x > 0 \\ 3.1 & \text{otherwise} \end{cases}$. See Figure 1.

$\mathbf{x}^{\text{opt}}$ optimal solution vector, such that $f(\mathbf{x}^{\text{opt}})$ is minimal.

Fig. 3.7 BBOB Notation from [222]

3.5.1 Seperable Function - Rastrigin

The Rastrigin problem has been designed to be a separable problem, meaning that the objective function can be decomposed into the sum of independent sub-functions, each dependent on a single variable. This may lead to an easier to solve optimisation problem however this does introduce some disadvantages including a large number of equally spaced local optima. The function in 3 dimensions can be seen in Figure 3.8a and its function shown in Equation (3.18).

$$f(\mathbf{x}) = An + \sum_{i=1}^{n}\left[x_i^2 - A\cos(2\pi x_i)\right] \text{ where } A = 10, x_i \in [-5.12, 5.12]. \tag{3.18}$$

3.5.2 Unimodal with High Conditioning - Bent Cigar

The Bend Cigar function, shown in Figure 3.8b is an example of a Unimodal with High Conditioning problem. This means that it has only one global optimal solution. This characteristic arises from the significant differences in variable scales, referred to as high conditioning, with the first variable representing a significantly larger impact than the others due to this scaling. The problem function is shown in Equation (3.19)

$$f(x) = z_1^2 + 10^6 \sum_{i=2}^{D} z_i^2 + f_{\text{opt}} \mathbf{z} = \mathbf{R} T_{\text{asy}}^{0.5}(\mathbf{R}(x - x^{\text{opt}})) \tag{3.19}$$

3.5.3 Multi-Modal with Global Structure - Multi-Modal Rastrigin

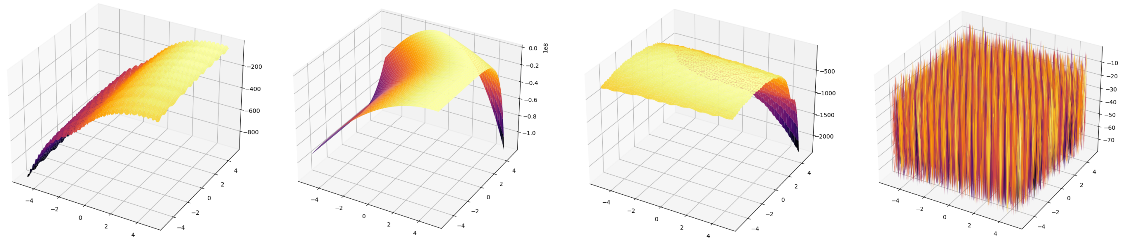
The Multi-Modal Rastrigin, seen in Equation (3.20), function shows an example of a function with more than one possible globally optimal solution while also having a large number of local optima. Figure 3.8c shows the problem in 3 dimensions. A key feature of this problem is that it contains global structure – in this case, this means that the optima are arranged with equal spacing, something that some metaheuristics may be able to learn and exploit.

$$f(x) = 10\left(D - \sum_{i=1}^{D} \cos 2\pi z_i\right) + ||z||^2 + f_{\text{opt}} \tag{3.20}$$

3.5.4 Multi-Model with Weak Global Structure - Katsuura

The Katsuura problem shown here is an example of a very complex problem landscape, as can be seen in Figure 3.8d. Here, we see that there are a significant number of local optima spread across the entire landscape. The problem has also been defined to have very low global structure for any algorithm to detect and exploit, meaning that in order to find a true globally optimal solution, an algorithm will need to both search globally and consider local exploitation to escape the very large number of local basins of attraction. Its function is shown in Equation (3.21)

$$f(x) = \frac{10}{D^2} \prod_{i=1}^{D} \left(1 + i \sum_{j=1}^{32} \frac{|2^j z_i - [2^j z_i]|}{2^j}\right)^{\frac{10}{D^{1.2}}} - \frac{10}{D^{1.2}} + f_{\text{pen}}(x) + f_{\text{opt}} \tag{3.21}$$



(a) Rastrigrin (b) Bent Cigar (c) Multi-Modal Rast. (d) Katsuura

Fig. 3.8 3D Landscapes of Selected BBOB Problems

Chapter 4

The Mathematics of Trajectory Mining

4.1 Data Mining

In this chapter, we define the methods used in the remainder of this thesis to perform the necessary data mining of EA search trajectories. Introduced in this chapter is a novel metric for measuring population diversity based on vector similarities - an approach that takes advantage of the linear nature of the subspace created by our selected decomposition method, PCA. By choosing PCA, as detailed in Chapter 3, we can analyse the search trajectories of binary, real-value, and nominal representations by measuring a variable's contribution to each resulting hyperplane at different stages of the search. Also covered in this chapter are the methods used to generate complementary sets of results when measuring a variable's influence on fitness using analysis of variance. We provide an overview of the ranked biased overlap technique for comparing ranked sets of results. The introduction of this method to our analysis has allowed us to generate explanatory sets of variable importances, ranked against each other, to highlight the most and least influential variables and sets of variables. By employing this method, we can compare the results of separate analysis techniques such as our vector-based variable contributions and the output of an analysis of variance test by comparing the resulting rankings.

4.1.1 Inter-Centroid Angle and Information Gain

Sub-Space Angular Metrics

In EAs, each generation is comprised of a set of solutions that represents the current position of the algorithm as it traverses the search space. To gain a better understanding of how this overall position changes over time and what information this may contain, we propose the following method to generate a novel set of angular-based metrics.

The first step in this process is the projection of the EA search trajectory, representing an optimisation run, into the resulting subspace generated via the PCA decomposition of the data. To achieve this, as noted in Chapter 3, first we apply SVD to the standardised search trajectory T, resulting in the three core matrices of left singular vectors, diagonal singular values and right singular vectors (principal components) as shown in Equation (4.1).

$$\begin{aligned} \mathbf{T} &= \mathbf{U}\Sigma\mathbf{P}^T \\ \widetilde{T} &= TP \end{aligned} \tag{4.1}$$

Here, $\mathbf{U}$ represents the left singular vectors, Σ is the diagonal matrix and $\mathbf{P}^T$ is the principal components. This can also be accomplished via the decomposition of S, the covariance matrix of X, to get the resulting eigenvectors and eigenvalues. By doing so, the resulting components (P) can be used as in Equation (3.12) to project the trajectory into the resulting subspace.

The second step, post-projection, is to generalise the algorithm's position concerning solutions contained in each generation of the trajectory. This is achieved by reducing the population of solutions to a single representative point, the "centroid" – the mean position across all components, calculated across all solutions in a generation. This process results in a single vector rather than a collection of vectors that represents the overall position of an algorithm at any given generation. This process, in which the centroid C at generation g with a population size of N is calculated in Equation (4.2).

$$C_g = \frac{1}{N}\sum_{i=1}^{N} \widetilde{x}_g^i \tag{4.2}$$

When working with PCA-derived subspaces, the angle between a solution and the origin of the space across an axis, in this case, a principal component, is used to measure the correlation of that solution with that component. The smaller the absolute value of the angle, the stronger the correlation as this indicates that the solution is well-represented by that component. The inverse is also true. This is because, as shown in Figure 4.1, the smaller the angle between two vectors the more similar they are. Here, the angle between vectors $\overrightarrow{A}$ and $\overrightarrow{B}$ can be used to measure their similarity. In the context of principal components, it can also be used to show that the smaller the angle between the vector representing a solution and the vector representing any given axis in the space, the greater degree to which the principal component captures the information or variance of that data point, such as the angle between vector $\overrightarrow{A}$ and the axis PC2.

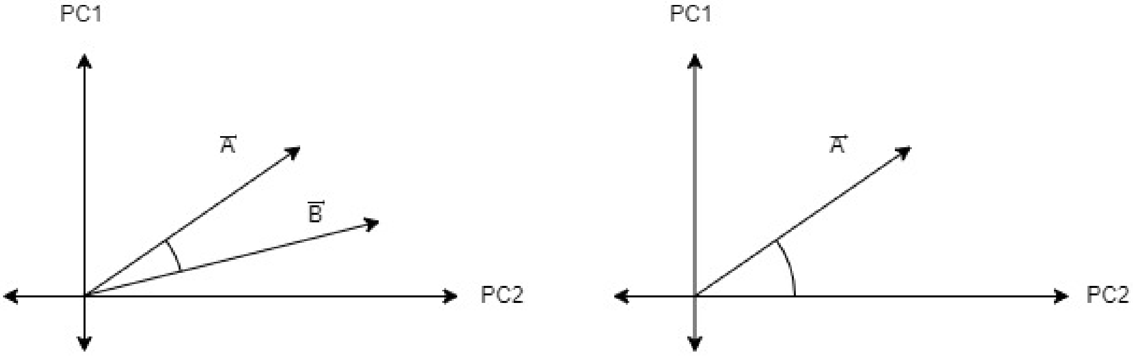


Fig. 4.1 Vector Cosine Similarity

Shown in Equation (4.3a) is how the cosine similarity between two vectors is calculated and the adaptation to using the arcoss value used in these experiments. The arcoss was selected in this metric as it converts the cosine similarity [-1,1] back into radians, providing a more intuitive indication of any geometric relationships between the two vectors. This conversion, as shown in Equation (4.3b), allows for the similarity measurement to span the range $[0,\pi]$, with a value of 0 indicating no differences between the vectors, and π indicating the maximal difference.

$$\text{Cosine Similarity} = \frac{\overrightarrow{A} \cdot \overrightarrow{B}}{\left\|\overrightarrow{A}\right\| \left\|\overrightarrow{B}\right\|} \tag{4.3a}$$

$$\alpha = \arccos\left(\frac{\overrightarrow{C_0} \cdot \overrightarrow{C_g}}{\left\|\overrightarrow{C_0}\right\| \left\|\overrightarrow{C_g}\right\|}\right) \tag{4.3b}$$

$$\overrightarrow{C_0}, \overrightarrow{C_g} = \text{Cluster Centroids (x, y, z)} \tag{4.3c}$$

Once each generation in the optimisation run has had its centroid calculated, we can use the adapted cosine similarity metric to generate two specific measures within the trajectory. The angle from the trajectory origin measures the angle between the centroid of the initial starting population in the trajectory and each subsequent population that was created. It is also possible to calculate the angle between adjacent centroids by altering Equation (4.3c) using C_g and C_{g+1}, where ($g <= \mathit{numGens}$). This allows for the angle between consecutive populations to be calculated. This results in two metrics, highlighting the overall similarity or difference from the initial starting population that each subsequent set of solutions, or the incremental changes in similarity as the search progresses and during convergence.

The centroids used in this process are shown in Figure 4.2. Here, the upper left diagram shows the comparison of each subsequent centroid to the origin ($\overrightarrow{C_0}$), used to measure the "Global" change in angle. The angle being measured is demonstrated in the lower left diagram. In the upper right diagram we show that to measure the change in angle for the "Local" metric, we compare each centroid to the next. The angles being measured in this scenario are demonstrated in the lower right diagram.

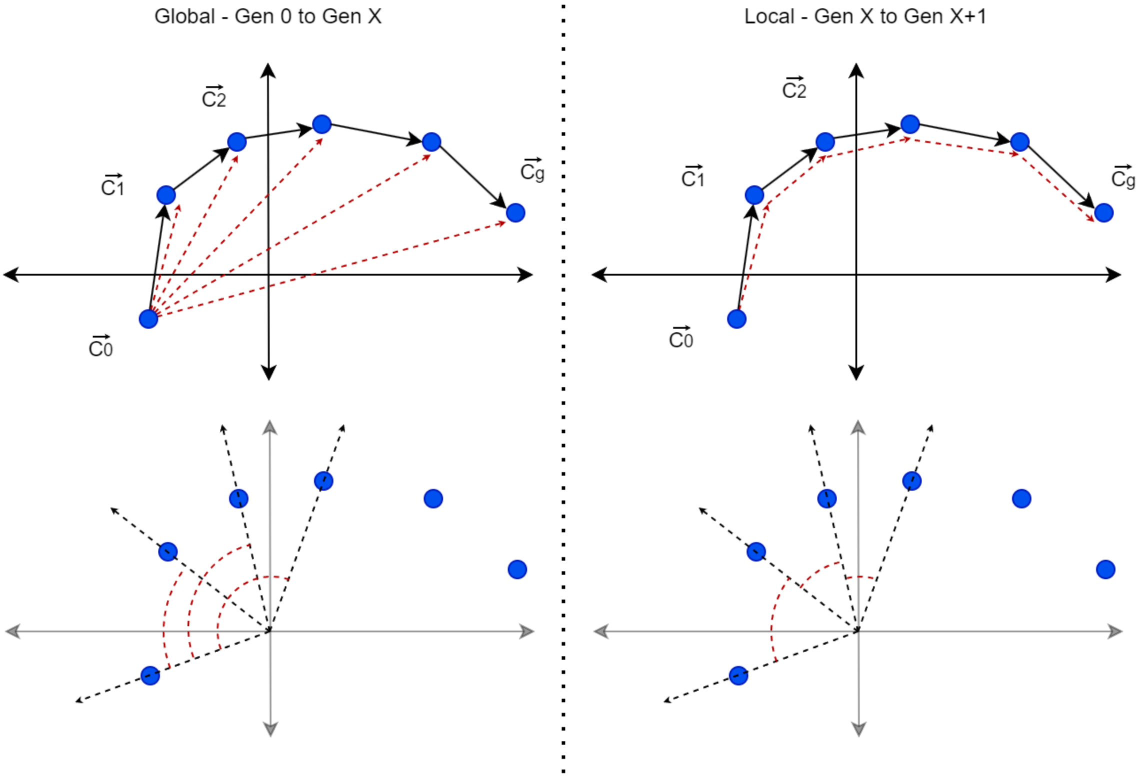


Fig. 4.2 Global and Local Vector Angles

It was then necessary to compare the results of this novel approach with a method known to be able to create a metric showing overall population diversity change in a singular value. This would allow the assessment of whether this novel method could retain enough geometric information post-decomposition and centroid-transformation, to be usable as a method of feature generation based on the algorithm's implicit estimation of the fitness function. It was determined that the Kullback-Liebleier Entropic Divergence [223] would be a suitable comparative metric. This metric was selected for comparison due its previous use in [224] in which it had already been demonstrated as a suitable metric for a range of problems. This provided a proven metric to compare to those being developed in this thesis as well as providing the opportunity to test whether our metrics were capable of detecting the same algorithm behaviours. Both KLd and the methods in this thesis use the same generations of solutions and both aim to measure the difference one generation and another. The KLd achieves this through the use of marginal probability distributions and the angular-based metrics in this thesis are designed to measure the difference between two vector representations of a generation of solutions via their cosine similarity. This provides the opportunity to compare two differing analysis techniques to verify whether those developed in this thesis are capable of capturing a similar level of information and behavioural differences between algorithms on a set of optimisation problems.

Kullback-Leibler Entropic Divergence

The Kullback-Leibler Entropic Divergence (KLd) metric is a method of measuring the differences between two probability distributions. Show in Equation (4.4), this process can be used to also measure the relative entropy between two populations of solutions generated by an EA.

$$KL_{\mathrm{d}}(Pr \parallel Q) = \sum_{x \in X} Pr(x) \log \left(\frac{Pr^{(t)}(x)}{Q^{(t_0)}(x)} \right) \tag{4.4}$$

Where P and Q are vectors of marginal probabilities for two different populations in the trajectory [225]. Using this, the "Information Gain", or difference in entropy, can be calculated. An example usage of this would be the Information Gain between the initial population and any other population of solutions - $Q(x)$ remains constant as the probability vector of the initial starting generation at t=0. This is commonly referred to as the "Global Learning" or "Global Information" metric. This value is expected to increase over time until a "steady state" is achieved, usually considered the point of convergence of an EA search.

Alternatively, as shown in [224], the equation can also be used to calculate the "Local Learning" or "Local Information Gain" value. This metric is a measure of the relative entropy between two consecutive populations of solution in the search trajectory, where $Q^{(i)}(x)$ and $P^{(i+1)}(x)$ are used instead. The behaviour of this metric is to increase over time however, as the remaining population diversity is reduced and the population of solutions converges, the Local Information Gain values should show a bell curve or peak. Noted in [224], it was the author's opinion that the peak of this bell-curve behaviour may represent the transition between a predominantly exploration-based search to a more exploitation-based search, using the implicitly gained knowledge of the problem structure to improve the overall solution quality, driving down the diversity between successive populations. Both approaches result in a value in the range [0,∞] with 0 indicating the two distributions are identical. Larger values indicate a substantial difference between the two distributions.

4.1.2 Variable Contributions

Contribution to Solution Projection

The inter-centroid angle metric defined earlier requires that a population of solutions be represented as a single vector in the created subspace. This process generalises the position of the algorithm by calculating the geometric centre of each generation as a point of reference against other generations of solutions. This allows us to consider the overall behaviour of the algorithm. An alternative approach is to instead measure the contribution each variable has in determining that position in the first place. The method of feature extraction presented is based on the concept of variable contributions – a measure of the contribution each variable has in determining a solution's position at a given generation. The process is shown in Equation (4.5).

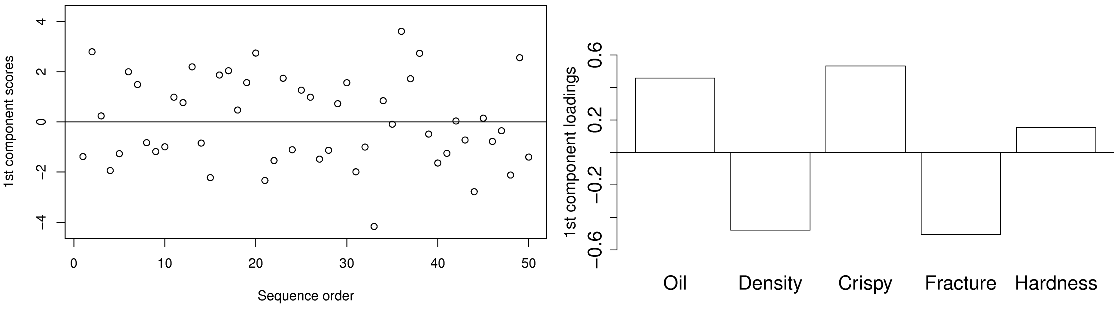
$$C_i' = \sum_{k=1}^{N} \left(\frac{\left(\sum_{j=1}^{n} p_j^i x_j^k \right)}{N} \right) p^i. \tag{4.5}$$

Here, C_i' belonging to $\mathbb{R}$, is a vector of mean variable contributions across any given component p^i of a generation of solutions with size N and dimensionality n. For each solution within that generation, we calculate this value for each of the variables and take the mean across all solutions in the same generation. This provides us with a method for describing the influence each variable in the problem has in determining the current populations' overall position in the subspace across component p^i.

Proportional Variable Alignment to Changing Fitness

The subspace generated by the application of PCA to a search trajectory can be used to help describe the overall structure of said trajectory. The coefficients of each resulting axis define the components and it is these coefficients that can help gain a better understanding of the data. A solution's score on a given PC is the projection of that solution into the new coordinate system, in this case, the generated subspace. Here, we define the score of a solution as the distance from the origin to the perpendicular projection across a given PC.

The component coefficients, sometimes referred to as "Loadings" or "Weightings" depending on the context, convey the relationship between the variables in a solution and that component itself. A solution's score indicates its distance from the origin along a given component, outlining its projection. The larger the absolute score value, the stronger that projection. As PCA uses means-centred data, the resulting component coefficients in a high-score solution better reflects the relationships in the original data. If a solution has a high positive score across a component, the sign and magnitude of the coefficients will tend to correspond with whether the variable values of that solution lie above or below the mean of the mean-centred data. An excellent example of this can be seen in [226] in which a food texture dataset is analysed using PCA. This work highlights the relationship between a solution's score, a component's coefficients and the variable values in the solution itself. Shown in Figures 4.3a and 4.3b is an extract from that work.



(a) Food Data PC1 Scores

(b) Food Data PC Coefficients

Fig. 4.3 Food Data PC Projection Results From [226]

Here, the figures show the resulting solution scores across the first component for the PCA results in Figure 4.3a and Figure 4.3b shows the loadings of the first component. As noted in the work, sample 36 has a high positive score of 3.6 and has variable values [21.2, 2570, 14, 13, 105] and 1st component loadings of [+0.46, -0.47, +0.53, -0.50, 0.15]. In the dataset, these values would be considered correctly described by the loadings of the first component in terms of being either higher or lower than the mean value in the original data - positive values suggest above the mean and negative suggest below. Sample 33 in the dataset has a strongly negative score of -4.2 and variable values of [15.5, 3125, 7, 33, 92]. Similarly, it is correctly described by the inverse of the loadings. These results are shown in Table 4.1.

Variable	Mean	SD	Sample 36	Sample 33
Oil	17.2	1.59	21.2	15.5
Density	2857.6	124.5	2570	3125
Crispy	11.52	1.78	14	7
Fracture	20.86	5.47	13	33
Hardness	128.18	31.13	105	92

Table 4.1 Food Data Variable Information from [226]

In this thesis, we define variable alignment at any given generation as the correct description of a variable by a component in relation to that component's coefficients. These descriptions can be seen in Table 4.2.

	+ve Score	-ve Score
+ve Loading	Above Mean	Below Mean
-ve Loading	Below Mean	Above Mean

Table 4.2 Score to Loading Alignment Chart

For sample 36 in the example given, 4 out of the 5 variables would be considered aligned - Oil, Density, Crispy and Fracture. As Hardness has a small absolute value across the first component it is less likely to be aligned however, as shown in their work, the second principal component has a higher loading for that variable, meaning that it is more likely to be better described by that component than the first.

The direction of projection may align either in the same or opposite direction as the component itself, which could be positively or negatively oriented concerning fitness improvement. An example of this is shown in Figure 4.4A, 4.4B, and 4.4C. We know that the direction of travel of the algorithms Search Trajectories (T) points from an initially worse fitness to better fitness sets of solutions, represented in Figure 4.4C.

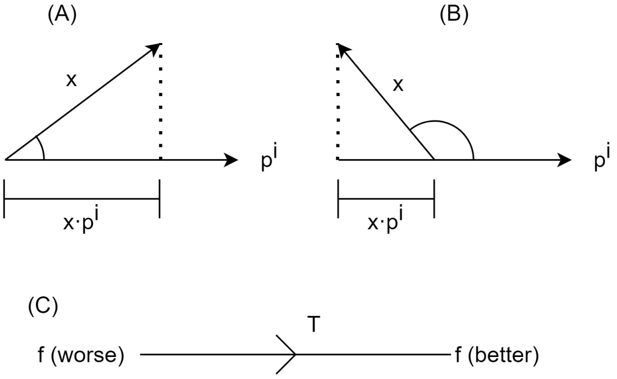


Fig. 4.4 Solution Alignment to PC

The PCs may be positively or negatively orientated with improving fitness depending on the results of the decomposition of T. During a defined search trajectory window, which consists of a set of adjacent generations, we consider variable alignment to fitness change as the detectable presence of a consistent alignment of variables throughout that specific window. This allows us to attribute the majority of solution quality improvements in a given window to highly aligned and highly contributory variables in the solutions of that window.

$$V_j^i(x) = \left\{ \begin{matrix} 1 \text{ if} & x \cdot p^i p_j^i \geq 0 \\ -1 & \text{Otherwise} \end{matrix} \right\} \quad (4.6) \quad G_i(x_j) = \left\{ \begin{matrix} 1 \text{ if} & (V_j^i(x) \geq 0 \text{ and } x_j \geq \bar{T}_j) \\ 1 \text{ if} & (V_j^i(x) \leq 0 \text{ and } x_j \leq \bar{T}_j) \\ 0 & \text{Otherwise} \end{matrix} \right\} \tag{4.7}$$

We can identify if a variable in a solution aligns with better fitness based on its projection onto PCs. Shown in Equation (4.6), this determines the sign of each variable in solution x based on its score $(x \cdot p^i)$ across PC p_i. Then, we compare each variable value x_j with its mean value across the trajectory T_j as shown in Equation (4.7). As the components were calculated from the scaled and means-centred data, a positive x_j will be above the mean and a negative will be below. This process results in a binary vector indicating whether this is true or false for each variable in the solution. This process is repeated across all components for all variables in a solution and the mean value per component is taken. Equation (4.8) shows how we calculate a total value for each solution in terms of the number of variables aligned to any given component. This is to provide a single value representing the proportion of variables aligned in a generation of solutions.

$$PG_i = \frac{\sum_{k=1}^{N} G_i(x^k)}{N} \tag{4.8}$$

Where PG_i is a vector representing the proportion of times that G_i in P_i for a generation of solutions was true across all N dimensions

4.1.3 Multiple Correspondence Analysis Variable Contributions

As shown in Chapter 2, it is possible to link MCA to PCA such that the application of an un-standardized PCA to an indicator matrix such as a Transformed Complete Disjunctive Table (TCDT), can lead to the same results as MCA. The first step in this process is the creation of the Complete Disjunctive Table (CDT), where categorical variables are converted into one-hot encoded dummy variables. The CDT values, denoted as x_{ik}, are transformed using y_{ik}, the proportion of solutions containing that value, as shown in Equation (4.9) [227]:

$$x_{ik} = y_{ik}/P_k - 1 \tag{4.9}$$

Applying PCA to the transformed CDT (TCDT) allows us to extract directional vectors from the trajectories by projecting the values into a lower-dimensional Euclidean space. As outlined in Equation (4.10), PCA produces m orthonormal eigenvectors in $\mathbb{R}^n$, each of size $n \times 1$. The components of these vectors, $[p_1^i, \ldots, p_n^i]$, represent the contribution of each variable to the respective principal component, maximizing variance in the dataset.

$$\begin{aligned} &P = [p^1, \ldots, p^m], m \leq n \\ &p^i = [p^i_1, \ldots, p^i_n] \end{aligned} \tag{4.10}$$

Using this method to generate the required subspace allows the extension of these methods to nominal value solution representations, such as those in the Twelve Week Minimally Disruptive Roster Allocation problem outlined later in Chapter 7. The Mean Squared Cosine (MSC) for each variable is calculated using Equation (4.11), (4.12), and (4.13), where $PC_{x_{nc}}$ is the principal coordinate of the variable x_n in category c, PC_{x_n} is the principal coordinate across all categories, λ_m is the eigenvalue of component p^m, and the factor loadings are Factor Loadings(x_n, p^m). The MSC value measures the variance captured by each category in each variable in the MCA space, indicating the significance of categories in the MCA analysis.

$$PC_{x_{nc}} = \sqrt{\lambda_m} \cdot \text{Factor Loadings}(x_n, p^m) \tag{4.11}$$

$$PC_{x_n} = \sqrt{\sum_{c=1}^{num_c} \lambda_m \cdot \text{Factor Loadings}(x_n, p^m)} \tag{4.12}$$

$$\text{MSC}(x_n) = \frac{1}{num_c} \sum_{c=1}^{num_c} \left(\frac{PC_{x_{nc}}}{PC_{x_n}} \right)^2 \tag{4.13}$$

The strength of the relationship between the observed categories and the given variable in the MCA-derived subspace is measured by the MSC - the higher the value, the stronger the relationship. This in turn implies the greater importance of those categories in capturing the structure and variability in the MCA analysis.

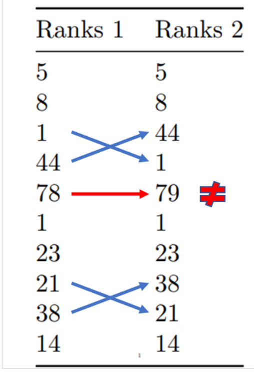
4.2 Comparative Techniques

As noted in Chapter 2.3, we aim to use the techniques in this thesis to generate a set of interpretable results for both the User and the Developer. It is presumed that the Developer will have a higher level of understanding of the principles discussed in this thesis however a method of conveying our findings in the form of an interpretable explanation to the User is needed. To achieve this, it was decided that presenting the User with a list of variables, ranked by their relative influence, would constitute an interpretable explanation regarding variable importance. To achieve this we implemented the Weighted Ranked Biased Overlap ranking method. This would allow us to rank and compare results from more than one analysis method while maintaining a level of interpretability by presenting the results from highest to lowest influence to solution fitness to the User.

As PCA is a variance-based method of decomposition in which each hyperplane generated is calculated such that it maximises variance in the data, it was necessary to select a suitable method of result validation. To this end, it was decided that Analysis of Variance (ANOVA) would suit our needs as it is similarly variance based and designed to compare the difference between two sets of means for statistical significance. As we are using solutions to an optimisation problem, it is possible to use ANOVA to measure the difference in means between a set of solutions and a dependant variable, in this case the set of solution fitnesses, to measure whether there is a statistically significant relationship between the two.

4.2.1 WRBO

Weighted Rank Biased Overlap (WRBO) [228] is a method of calculating a similarity score between two lists of ranked items. The process generates a similarity score taking a value of $[0,1]$, where 1 indicates a complete overlap between the two lists in both rank order and members and 0 when there is no match on either, demonstrated in Figure 4.5.



Ranks 1	Ranks 2
5	5
8	8
1	44
44	1
78	79 ≠
1	1
23	23
21	38
38	21
14	14

WRBO Similarity = 0.802 (p=0.9)

Fig. 4.5 WRBO List Comparison Example

There are other methods of measuring the similarity between lists, with the Spearman Rank Correlation [229] and the Kendall Tau [230] method being examples. WRBO however has two significant advantages concerning the work done during this project. Firstly, unlike either Spearman or Kendall Tau, the WRBO approach does not need both lists to be of the same length and have the same elements. This is useful when comparing subsets of variables against each other, such as any Top-N lists containing variables ranked by another metric. In that case, there is no guarantee that the lists will contain the same variables. Secondly, the

WRBO method has been designed to allow the use of an internal parameter “wp” as shown in the Algorithm 7.

Algorithm 7 Rank Biased Overlap (RBO) Pseudo-Code

```
Require: Two lists $S$ and $T$, weight parameter $wp$ (default: 0.9)
Determine the maximum length $k \leftarrow \max(\text{len}(S), \text{len}(T))$
Calculate the intersection at depth k: $x_k \leftarrow |\text{set}(S) \cap \text{set}(T)|$
Initialize summation term: summ_term $\leftarrow 0$
for $d = 1$ to $k$ do
    Create sets from the lists:
    set1 $\leftarrow$ set($S[:d]$) if $d < \text{len}(S)$ else set($S$)
    set2 $\leftarrow$ set($T[:d]$) if $d < \text{len}(T)$ else set($T$)
    Calculate intersection at depth d: $x_d \leftarrow |\text{set1} \cap \text{set2}|$
    Compute agreement at depth d: $a_d \leftarrow \frac{x_d}{d}$
    Update: summ_term $\leftarrow$ summ_term $+ wp^d \cdot a_d$
end for
Calculate Rank Biased Overlap (extrapolated):
$rbo\_ext \leftarrow \frac{x_k}{k} \cdot wp^k + \frac{(1-wp)}{wp} \cdot$ summ_term =0
```

As noted in [231] "...RBO solves for the 3 drawbacks observed in Kendall Tau by using weights for each rank position. The weights are derived from a convergent series ...". The three drawbacks mentioned are:

- “It requires the two ranking lists to be conjoint (same elements in both lists)”
- “It is unweighted i.e., it places as much emphasis on the disagreement at the bottom of the list as much as the top (in popular search engines, the results in the “head” matter a lot more than the results in the “tail”)”
- “The contribution of a single discordant pair decreases with the increase in the depth of the ranking i.e., τ values are intrinsically linked to the depth of the ranking list.”

Equation (4.14) shows the method for how the similarity score is calculated for two infinite-length sets.

$$RBO(S,T,wp) = (1-wp)\sum p^{d-1} \cdot A_d \tag{4.14}$$

$$W_{RBO[1:d]} = 1 - wp^{d-1} + ((1-wp)/wp) * d * (ln(1/(1-wp)) - \sum(wp^i)/i) \tag{4.15}$$

Here, $d = 1$ to ∞ (depth of the ranking being examined), $X_d = |S_{:d} \cap T_{:d}|$ which is the size of the overlap of S and T to depth d and $A_d = X_d/d$ is the agreement between S and T given by the proportion of the size of the overlap to depth d, as per the original definition. Equation (4.15) shows the author's method of how the overall weight of the top d variables, or ranks, contribute to the similarity metric score, where $i = 1$ to $d - 1$.

Here, the parameter wp can be used to allocate a disproportionate weighting to a subset of the lists used in the comparison when generating the similarity score. This value is calculated based on the number of top values in the list you wish to allocate the higher impact. As an example, should we wish the top 10 items in a list to have a higher contribution, a value of $wp = 0.9$ would mean that the top 10 variables contribute a total of 86% of the similarity score.

4.2.2 Analysis of Variance (ANOVA)

To gain a better understanding of the trajectory analysis results, we employ Analysis of Variance (ANOVA), which is a statistical method for the comparison of means across multiple groups. This is used to generate a comparative set of variable rankings. In our datasets, we can use ANOVA to compare the means of each variable in our solutions to the dependent variable - solution fitness. This analysis technique is used to detect whether there is a relationship between each variable and the fitness of a solution. This is done by using the sum of squares between value groups and the sum of squares within groups. The resulting *p-value* can be used to indicate whether any detected relationship is statistically significant. For this thesis, we consider all variables to be independent. This decision means that we can apply the ANOVA test to each variable-fitness pair separately and calculate the "partial eta squared" value for each pair. Partial eta squared is a measure of effect size in ANOVA that represents the proportion of total variance that is explained by an effect while controlling for other effects. To calculate the partial eta squared values, we use the Python "statsmodel" package [232] implementation of ANOVA. Equations (4.16) to (4.18) show how we calculate the partial eta value (η_p^2). Here, k is the number of solutions in our trajectory (gN), n is the number of variables, and x_{ij} is the jth variable in the solution i of the whole trajectory. The mean value of all variables in solution i is shown as $\bar{x}_i$.

$$SS_{\text{within}} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)^2 \tag{4.16}$$

$$SS_{\text{between}} = \sum_{i=1}^{k} n_i (\bar{x}_i - \bar{x})^2 \tag{4.17}$$

$$\eta_p^2 = \frac{SS_{\text{between}}}{SS_{\text{between}} + SS_{\text{within}}} \tag{4.18}$$

We use the value $\bar{x}_i$, in conjunction with the resulting *p-value* generated for each variable, to determine the influence that the variable x_i has on Fitness and whether that influence is statistically significant. For this thesis, if $p < 0.05$, we reject the null hypothesis that x_i has no measurable influence on fitness across all solutions in a trajectory. Once complete, we use the partial eta to rank each variable in terms of the size of their effect on the fitness measured.

Chapter 5

Trajectory Mining for Information Gain and Sub-Space Angular Metrics

5.1 Introduction

In this chapter, we examine the relationship between the geometrically sensitive features found in search trajectories and the observed diversity of the populations of solutions generated by population-based metaheuristics on binary-string problems. It is often observed that as these search methods converge on an optimal or near-optimal set of solutions, the diversity of the population of solutions tends to decrease. This is indicative of the algorithm's learning process about the fitness function and problem structure – aspects of the problem that are encoded within the populations of solutions. This typically takes one of two forms in EA – implicitly in the case of GAs or explicitly in the case of probabilistic models such as EDAs.

Various metrics exist for measuring population diversity in these algorithms. As reviewed in [233], these include the Hamming Distance, which calculates the pairwise variable differences between solutions, and the Moment of Inertia, detailed in [234]. Another method for measuring population diversity is the Kullback-Leibler Entropic Divergence (KL_{d}), which is based on Information Gain and Shannon's Entropy, as described in [235] and [224]. The creators of this method considered it useful for understanding the algorithm's transition from exploration to exploitation phases – two traits of EAs that generally describe their search behaviour as the problem space is explored.

Exploration refers to the algorithm's ability to investigate a wide range of the solution space. This process is crucial for discovering diverse and potentially promising areas that might contain optimal or near-optimal solutions. Effective exploration prevents the algorithm from converging prematurely on local optima, ensuring a thorough search of the solution space. Exploitation, on the other hand, involves the algorithm's focus on refining and improving solutions within a specific region of the solution space. Once promising areas are identified through exploration, exploitation works on fine-tuning these solutions to reach the optimal or near-optimal points. This process is essential for intensifying the search in areas with high potential, leading to the discovery of the best possible solutions.

Both of these traits of phases should however not be considered in isolation – It's important to clarify that these are not unrelated concepts. Excessive focus on exploitation can lead to premature convergence on local optima, whereas emphasising exploration too much can result in spending too much time on lower-quality solutions and disregarding valuable information already acquired.

By monitoring the change in population diversity throughout the optimisation runs, this chapter will explore the generation of a set of features capable of relating variable patterns or low-order variable interactions to the fitness function. To achieve this, we first explore whether structurally important geometric features are preserved during the decomposition process using PCA. To this end, we utilise a method based on the Cosine Similarity measurement as shown in Chapter 4 and is checked against the information-based population diversity measurement KL_{d}. Following this, we analyse the resulting component coefficients from the transformation of the search trajectories for explanatory features regarding problem structure.

A selection of binary benchmarking problems was selected for exploration in this chapter. These were selected for their relative simplicity and known problem structure. The techniques used in this chapter are not restricted in use to binary search spaces and are equally applicable to real-valued problems. However, as an initial investigation into their suitability, binary problems were selected. This allowed us to validate whether our initial approaches were able to detect whether low-order problem structure was being captured within the EA search trajectories generated. By using these problems as part of an initial investigation, the contributions of the work in this chapter address research questions (Q2) and (Q3).

We show that by using a formalised structure of solution-population representation, we can successfully mine features from the search trajectories of EAs using PCA. This is done by achieving research objectives (O2), (O3) and contributes towards the completion of objective (O5) in which we use the structure outlined in Chapter 3 and techniques outlined in Chapter 4. This results in the generation of features that aid in end-user understanding of the main drivers of solution quality to the algorithms used. The results and experimentation in this chapter are an extension of techniques published in [2].

5.2 Experimental Setup

The datasets used in this chapter are generated by solving a set of binary string problems representing both bi-variate and higher-order variable interactions. These problems, as outlined in Chapter 3, are the 1-D Checkerboard, Royal Road and Trap-5 problems. These were selected as they act as a representative set of binary-representation problems showing both low and high-order variable interactions for the EAs to detect during optimisation.

The algorithms selected for this work were a Genetic algorithm and a modified version of the Population-Based Incremental Learning (PBIL) Algorithm, as described in [236, 237] and outlined in Chapter 2. This modified PBIL variant incorporates a negative mutation rate and a mutation shift value. These algorithms were selected so that we could evaluate the performance of a univariate solver, the PBIL, and a more conventional GA, in their ability to detect problem structure and how this is represented in their respective search trajectories. Both represent relatively simplistic entries in the EA world, allowing us to focus on the fundamental behaviours of these algorithms on benchmarking problems with known optima and structure.

5.2.1 Problems

The 1D Checkerboard

As detailed earlier in Chapter 3.4.1, here we show the fitness function used to score solutions in the 1-D Checkerboard problem in Equation (5.1)

$$CHECK_{1D}^{l}(x) = \sum_{i=0}^{l-2} \left\{ \begin{matrix} 1, & x_i \neq x_{i+1} \\ 0, & x_i = x_{i+1} \end{matrix} \right\} \tag{5.1}$$

This problem has two possible global optima as noted in Chapter 3.4.1. There, we also show the structure of the problem - a chain structure in which each variable interacts with the adjacent variables with the optimal solution showing an alternation between "1" and "0".

The chain structure is cyclical - the implementation used checks for this interaction between the first and last allele in the solution, allowing for a fitness value that is equal to the length of the bit-string, provided the length is an even value. For these experiments, a bit string length of 40 is used, meaning that an optimal solution will result in a fitness value of 40.

The Royal Road

The implementation of the Royal Road function used in this chapter is fully outlined in Chapter 3.4.2 and the definition is repeated in Equation (5.2). Here, a set of schema have been defined for which the fitness is only gained when all members of that schema are assigned the value of 1. In this implementation, with a bit string length of 40, a total of 8 schemas were defined. Each schema is composed of 5 adjacent bits in the solution, with no overlapping. The first schema would contain the members of the solution x_1 to x_5, the second would contain x_6 to x_{10} and so forth.

$$R_1(x) = \sum_{i=1}^{5} \delta_i(x) o(s_i), \text{ where } \delta_i(x) = \left\{ \begin{matrix} 1 & \text{if } x \in s_i \\ 0 & \text{otherwise} \end{matrix} \right\} \tag{5.2}$$

The Trap-5

Finally, as also fully outlined in Chapter 3.4.3, we consider the Trap-5 problem. This is an iteration of the Trap-K problem in which the schema size has been set to 5. This problem shares the same schema pattern as the previously outlined Royal Road problem, with each consisting of a non-overlapping set of 5 consecutive bits in the bit string. The fitness function for this problem is repeated in Equation (5.3a) and (5.3b).

$$f(x) = \sum_{i=1}^{n/k} trap_k(x_{b_{i+1}} + ... + x_{b_{i+k}}) \tag{5.3a}$$

$$trap_k(u) = \left\{ f_{\text{high}} \text{ if } u = k, f_{\text{low}} - u\frac{f_{\text{low}}}{k-1} \text{ otherwise} \right\} \tag{5.3b}$$

The Trap-5 problem used in this chapter is designed to be deceptive in that, the closer the bits in a schema get to being all 1's, the lower their impact on fitness is. This is overcome only when all bits in a schema are 1's, coercing the algorithm towards a less than optimal solution.

5.2.2 Algorithm Run Settings

Selection Methods

In GAs and other EAs the selection methods play a crucial role. They are responsible for selecting a subset of solutions from the current population to generate the subsequent population. It should be noted that the selection methods applied by these algorithms might contribute to the detection of variable dependencies in the results. To mitigate any potential bias arising from the use of a single selection method, three distinct selection methods were used, as shown in Table 5.1. This was done to test the effect that each selection method and its associated selection pressure has on the experimental results.

Selection Type	**Pool Size**	**Relative Selection Pressure**
Truncation 20%	Top 20% by Fitness	High
Truncation 50%	Top 50% by Fitness	Low
Tournament	Randomly Selected 5	High

Table 5.1 Selection Operators

These selection methods generally exhibit a preference for solutions with higher quality and therefore “better” fitness. Because of this, the detection of certain variable dependencies may be influenced by the behaviour of the selection method. To address this potential bias, the experiment varied the selection pressure exerted on high-fitness solutions. This was achieved by implementing both a top 50% and top 20% criteria in the truncation selection method, alongside a tournament size of 5 in the tournament selection method. This approach resulted in a dataset that represents a spectrum of selection pressures, thereby allowing for a more detailed analysis of the impact exerted by the selection operator in the experiments. The resulting set of all experimental runs for each algorithm is detailed in Table 5.2.

Algorithm Runs

Each algorithm was run on the set of defined problems to produce the trajectories for analysis. The problem length was set to 40 bits for all problems, considering the relative simplicity of the optimisation problems themselves. A population size of 100 was chosen to ensure a diverse selection pool, and each optimisation run spanned 100 generations. To extract meaningful features from a substantial volume of search trajectories, 100 optimisation runs were conducted for each algorithm-problem-selection combination. The main focus of this set of experiments was the extraction of features from the trajectories and not the fine-tuning of algorithm performance. To this end, no specific alterations of algorithm parameters were done beyond an initial trial to ensure that each algorithm was capable of solving the problems within the allotted generation limit.

The settings for the GA and PBIL runs are detailed in Table 5.3 which provides an overview of the parameter values used for both algorithms across all algorithm-problem pairs.

The GA used was the PYMOO [43] implementation and run settings were those recommended by the authors. Here, tournament selection was used to select two parent solutions

Algorithm	Problem	Selection	Pool Size	Relative Pressure
GA	1D Checker	Truncation 20%	Top 20% By Fitness	High
GA	1D Checker	Truncation 50%	Top 50% By Fitness	Low
GA	1D Checker	Tournament	Randomly Selected 5	High
GA	Royal Road	Truncation 20%	Top 20% By Fitness	High
GA	Royal Road	Truncation 50%	Top 50% By Fitness	Low
GA	Royal Road	Tournament	Randomly Selected 5	High
GA	Trap-5	Truncation 20%	Top 20% By Fitness	High
GA	Trap-5	Truncation 50%	Top 50% By Fitness	Low
GA	Trap-5	Tournament	Randomly Selected 5	High
PBIL	1D Checker	Truncation 20%	Top 20% By Fitness	High
PBIL	1D Checker	Truncation 50%	Top 50% By Fitness	Low
PBIL	1D Checker	Tournament	Randomly Selected 5	High
PBIL	Royal Road	Truncation 20%	Top 20% By Fitness	High
PBIL	Royal Road	Truncation 50%	Top 50% By Fitness	Low
PBIL	Royal Road	Tournament	Randomly Selected 5	High
PBIL	Trap-5	Truncation 20%	Top 20% By Fitness	High
PBIL	Trap-5	Truncation 50%	Top 50% By Fitness	Low
PBIL	Trap-5	Tournament	Randomly Selected 5	High

Table 5.2 Binary Problem Dataset Details

Alg	Pop Size	Length	Runs	MaxGen	mutRate	Sel.	Cross.	mutShift	l-Rate
GA	100	40	100	100	0.005	Tour & Trunc	1-Point	N/A	N/A
PBIL	100	40	100	100	0.005	Tour & Trunc	N/A	0.05	0.1

Table 5.3 Algorithm Run Specifications

randomly selected from the population. A single-point crossover operator was used to handle crossover. The mutation used was a polynomial [238] function, details of which can be found in [239]. As noted, the implementation and run settings of the PBIL were based on those found in [236, 237].

5.3 Analysis Methods

5.3.1 Inter-Cluster Angle Similarities

The inter-cluster angle similarities, as detailed in Chapter 4, are used to compare the similarity of both consecutive and subsequent generations of solutions in a search trajectory. The process by which these are calculated is repeated in Equation (5.4a) to (5.4c) for reference.

$$\text{Cosine Similarity} = \frac{\overrightarrow{A} \cdot \overrightarrow{B}}{\left\|\overrightarrow{A}\right\| \left\|\overrightarrow{B}\right\|} \tag{5.4a}$$

$$\alpha = \arccos\left(\frac{\overrightarrow{C_0} \cdot \overrightarrow{C_g}}{\left\|\overrightarrow{C_0}\right\| \left\|\overrightarrow{C_g}\right\|}\right) \tag{5.4b}$$

$$\overrightarrow{C_0}, \overrightarrow{C_g} = \text{Cluster Centroids (x, y, z)} \tag{5.4c}$$

Once all solutions in the optimisation run have been projected into the resulting subspace, the "centroid" is calculated for each generation. This process reduces each generation of solutions in an optimisation run to a single vector representing the overall position of the algorithm in the search space. The equations to do so are repeated in Equations (5.5a) to (5.5c) for reference.

$$T = U\Sigma P^T \tag{5.5a}$$

$$\widetilde{T} = TP \tag{5.5b}$$

$$C_g = \frac{1}{N}\sum_{i=1}^{N} \widetilde{x}_g^i \tag{5.5c}$$

Here, $\widetilde{T}$ is the projected trajectory or optimisation run after the principal components are calculated via SVD shown in Equation (5.5a). Each generation of projected solutions in $\widetilde{T}$ are then used to generate the set of centroid vectors C_g. This single vector is calculated by taking the mean position of each solution, across all variables in the first 3 components in a given generation. This mean position is calculated, resulting in the singular vector representation of the algorithm's position at a given generation in the subspace. These vectors are then used in the analysis to measure the similarities between adjacent generations and each generation to the first. This process is repeated for each optimisation run separately and the mean values are taken across all runs.

5.3.2 Entropic Divergence Population Diversity

The Kullback-Leibler Entropic Divergence (KLd) is used in this Chapter as a measure of population diversity, as outlined in Chapter 4. Within each generation of solutions, there exists a probability distribution for every variable. Repeated in Equation (5.6) for reference is the method of calculating the entropic divergence value between two marginal probability distributions.

$$KL_{\mathrm{d}}(P \parallel Q) = \sum_{x \in \mathscr{X}} P(x) \log \left(\frac{P^{(t)}(x)}{Q^{(t_0)}(x)} \right) \tag{5.6}$$

In this Chapter, this will be used to measure the relative entropy between two populations of solutions generated by both the GA and the PBIL. This will use the same generations as the angular-based cosine similarity metric used, in which for each optimisation run, all solutions in a given generation are used. This process however does not require the projection of the solutions and so will use the unaltered, original data in the search trajectories. As with the Inter-Cluster Angle values, this process is repeated for each optimisation run in isolation and the mean values across all runs are calculated.

5.4 Results

5.4.1 Algorithm Performance

Shown in Table 5.4 are the optimisation results of each algorithm-problem pair. The table shows the mean fitness value found in the final generation across all 100 runs and both the mean minimum and maximum fitness attained at the end of the runs. It can be seen that overall, the GA performs better on all problems than the univariate PBIL, achieving a higher mean fitness in all instances. The GA is also shown to have consistently found higher quality solutions by the final generation as seen in the higher maximum fitness values.

In only 2 cases - that of the Royal Road with Tournament selection and Royal Road with 20% Truncation, did the PBIL find the best possible solution resulting in the maximum possible fitness value of 40. The GA, however, consistently discovered these solutions by generation 100 in all but the Trap 5 problem, with 38 or 39 being the best achieved in those cases. It should be noted however that due to the nature of the Trap-5 problem, a score of 39 would indicate that one of the 5-bit schemas contained four 0's. These results are highlighted in Figure 5.1 in which we show side by side the performance of each algorithm, plotting each selection method to compare performance. The plots show the mean fitness per generation across all 100 runs, split by selection method.

Here, we see that across both algorithms and all problem instances, the lower selection pressure Truncation 50% method results in the lowest overall performance. The Truncation 20% and the Tournament selection have similar results, showing the same overall results and a higher rate of solution quality improvement throughout the 100 generations. The GA results

Algorithm	Selection	Problem	Mean F.	Min F.	Max F.
GA	Truncation 20%	1D Checker	37.02	30	40
GA	Truncation 50%	1D Checker	36.57	26	40
GA	Tournament	1D Checker	37.28	30	40
GA	Truncation 20%	Royal Road	36.93	15	40
GA	Truncation 50%	Royal Road	35.92	10	40
GA	Tournament	Royal Road	38.10	25	40
GA	Truncation 20%	Trap 5	35.92	25	39
GA	Truncation 50%	Trap 5	34.80	21	38
GA	Tournament	Trap 5	35.94	24	39
PBIL	Truncation 20%	1D Checker	34.11	26	38
PBIL	Truncation 50%	1D Checker	24.35	14	34
PBIL	Tournament	1D Checker	33.09	24	38
PBIL	Truncation 20%	Royal Road	25.85	10	40
PBIL	Truncation 50%	Royal Road	5.19	0	25
PBIL	Tournament	Royal Road	21.32	0	40
PBIL	Truncation 20%	Trap 5	31.81	24	35
PBIL	Truncation 50%	Trap 5	18.74	6	33
PBIL	Tournament	Trap 5	30.11	20	34

Table 5.4 Binary Problem Results by Algorithm and Selection Type

show, by generation 20, that the Truncation 20% and Tournament methods have achieved almost all fitness improvements, in contrast to the lower pressure method which does not match the fitness values found until around generation 50. Overall the results show that the GA outperforms the PBIL which does not come as a surprise as the PBIL is not capable of capturing higher-order variable interactions within the probability vector used to generate successive populations. The results do show that, over time, both algorithms are capable of finding higher-quality solutions to each of the problems, under varying selection pressures. This is highly important as it shows that, at some level, the populations of solutions generated by both algorithms contain some implicit or explicit knowledge of the underlying problem structures which is being exploited to find higher-quality solutions.

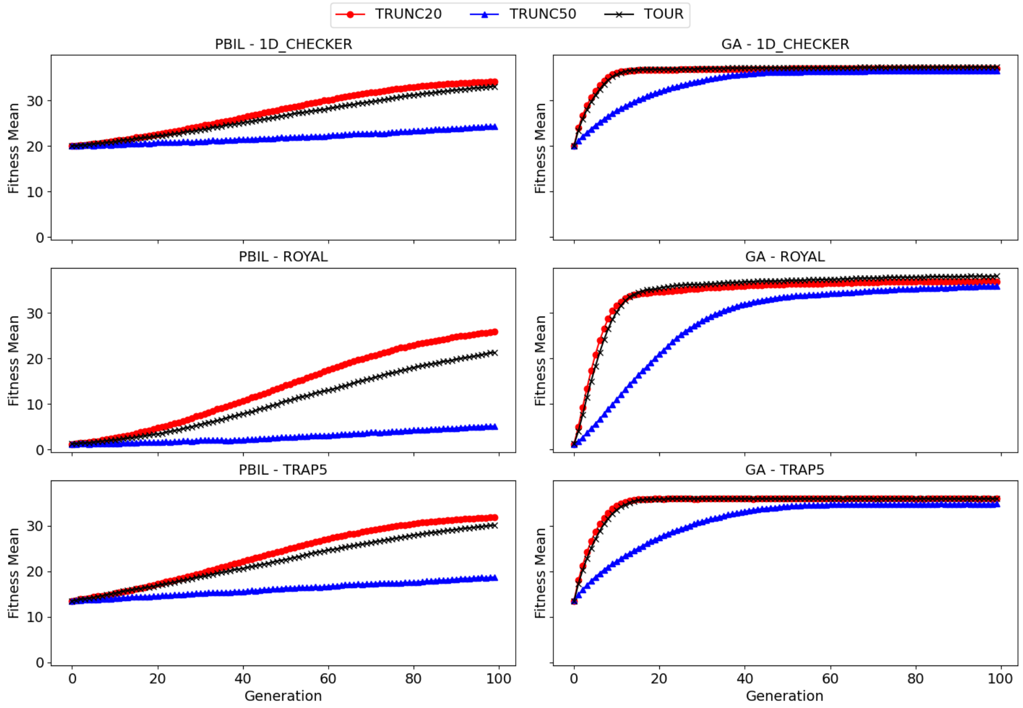


Fig. 5.1 Fitness Results by Selection Method and Problem

5.4.2 Decomposition Results

Shown in Table 5.5 is the resulting mean percentage of explained variance across the first 3 principal components for each algorithm-problem-selection combination. This is calculated by taking the mean across all of the 100 runs for each experiment set. This table shows that the GA components consistently describe a higher percentage of variation in the data than those of the PBIL, across all combinations. The highest found in the GA results were those of the Trap-5 with Truncation 20% which is surprising considering that the problem was designed to be intentionally deceptive to EAs. Similarly, the same combination for the PBIL also resulted in the highest total explained variance across the first 3 components out of all other tests. For both algorithms, a distinct pattern can be seen in the difference between the first and subsequent components. The first principal component explained a significantly larger level of variance in the data, with that percentage quickly dropping off as more components were added.

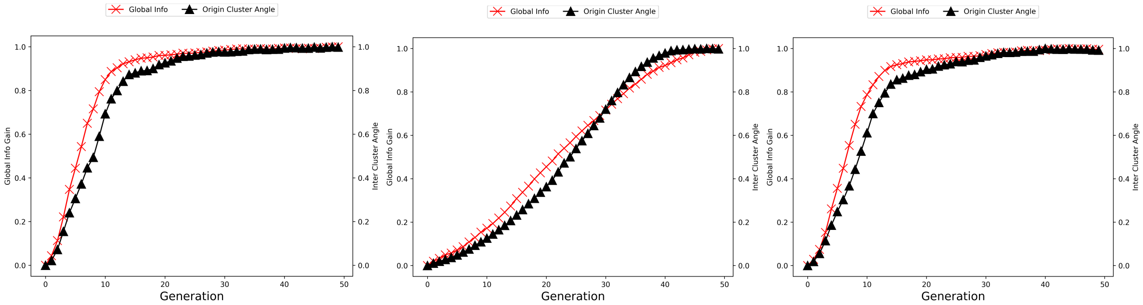
Algorithm	Problem	Selection	PC1 %	PC2 %	PC3 %	Total %
GA	1D Checker	Truncation 20%	29.1	8.1	5.8	43.0
GA	1D Checker	Truncation 50%	24.3	7.1	5.6	37.0
GA	1D Checker	Tournament	28.3	8.1	5.6	42.0
GA	Royal Road	Truncation 20%	26.9	7.4	5.2	39.5
GA	Royal Road	Truncation 50%	24.5	6.7	4.7	35.9
GA	Royal Road	Tournament	24.5	7.6	5.1	37.2
GA	Trap 5	Truncation 20%	33.2	8.1	5.6	46.9
GA	Trap 5	Truncation 50%	27.0	7.8	5.2	40.0
GA	Trap 5	Tournament	33.1	8.5	5.5	47.1
PBIL	1D Checker	Truncation 20%	18.4	5.1	3.3	26.8
PBIL	1D Checker	Truncation 50%	13.1	5.2	3.7	22.0
PBIL	1D Checker	Tournament	17.2	5.1	3.4	25.7
PBIL	Royal Road	Truncation 20%	18.0	5.5	3.4	26.9
PBIL	Royal Road	Truncation 50%	12.9	5.4	3.7	22.0
PBIL	Royal Road	Tournament	16.5	5.3	3.4	25.2
PBIL	Trap 5	Truncation 20%	19.9	5.4	3.3	28.6
PBIL	Trap 5	Truncation 50%	12.9	5.2	3.7	21.8
PBIL	Trap 5	Tournament	17.7	5.3	3.4	26.4

Table 5.5 PCA Mean Variance Explained by First Three Components

5.4.3 Information Gain and Vector Similarities

The results of the information gain and population angular results are shown in Figures 5.2 to 5.5 and A.1 to A.8. Here, we only show the results for the 1D Checkerboard problem as they are broadly similar to those of the Royal Road and Trap-5 problems and as such they can be found in Appendix A. The Figures show the mean information gain or relative entropy calculations plotted against the equivalent population vector results when using the same populations of solutions - this means that for each generation in an optimisation run, the values are calculated and the mean is taken across all 100 runs. The plots show the results scaled to [0,1] to allow for a clearer comparison of metric behaviour in each plot.

Figures 5.2 and 5.3 show the comparison of global information gain to the arcoss angle measurements between each population centroid and the first generation centroid for the GA and PBIL respectively, termed the "angle to origin". For both the GA and PBIL, both metrics have a highly consistent behaviour. This consistency is further highlighted in Table 5.6 which shows the correlation between these metrics. We show that, in the case of the GA, the level of global information gain and angle to origin metrics closely match the behaviour

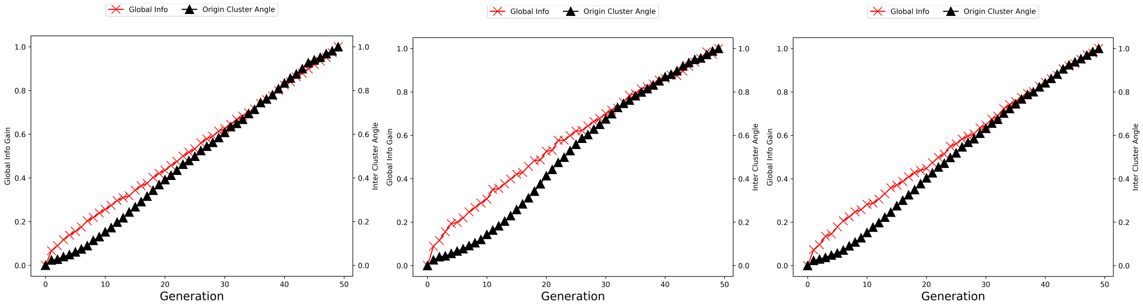


(a) Truncation 20%

(b) Truncation 50%

(c) Tournament

Fig. 5.2 Genetic Algorithm Mean Global Vs. Angle to Origin Results 1D Checkerboard by Selection



(a) Truncation 20%

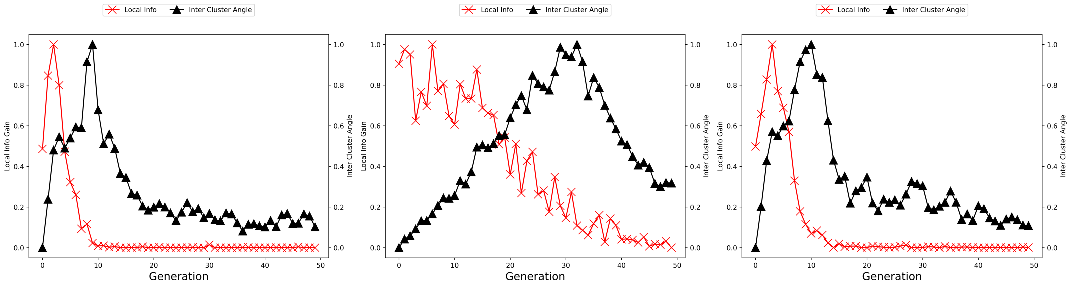
(b) Truncation 50%

(c) Tournament

Fig. 5.3 PBIL Mean Global Vs. Angle to Origin Results 1D Checkerboard by Selection

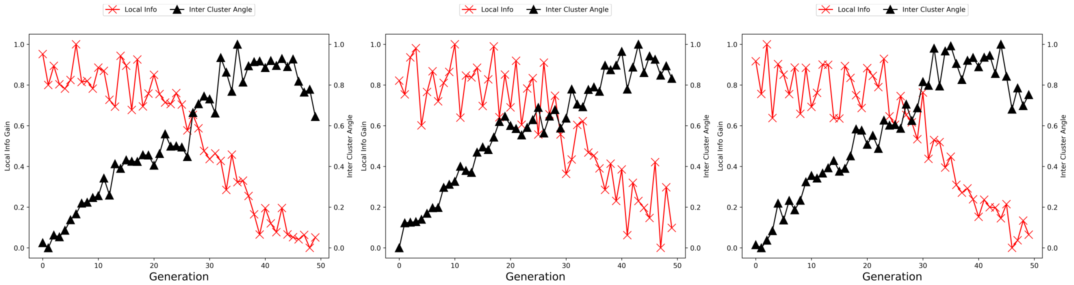
in the algorithm fitness performances seen in Figure 5.1. The Truncation 50% values for both metrics show a much slower, steady increase than the higher selection pressure methods which show a much higher initial rate. As outlined in Chapter 4, the higher the information gain value, the more dissimilar the two distributions are. As the values in both are scaled to $[0,1]$, we see that initially in both the Truncation 20% and Tournament selection, the rate of change between the first generation of solutions and each subsequent generation is high. At generation 20, we see that the rate of population diversity changes levels off. The PBIL results shown in Figure 5.3 show a more steady rate of change. This is also seen in the Fitness results in Figure 5.1 that shows a steadier, flatter rate of solution quality improvement. By the end of each optimisation run we see that both metrics are implying that the two population distributions - the first generation and the last generation - are significantly different. The rate of change between successive populations varies between algorithm and selection pressure however overall, both metrics are shown to observe the same behaviour.

The results of the local information gain and inter-centroid angle - a comparison between successive populations of solutions - are shown in Figures 5.4 and 5.5 for the GA and PBIL respectively. It is clear from these results that there is a greater level of variation between the algorithms and selection methods.



(a) Truncation 20% (b) Truncation 50% (c) Tournament

Fig. 5.4 GA Mean Local Vs. Inter-Centroid 1D Checkerboard by Selection



(a) Truncation 20% (b) Truncation 50% (c) Tournament

Fig. 5.5 PBIL Mean Local Vs. Inter-Centroid 1D Checkerboard by Selection

Here, we see that, in the case of the GA, both the Truncation 20% and Tournament selection methods result in a similar behaviour to the information gain however there is a distinct lag in the detection of the peak, of around 10 generations. The results for the Truncation 50% show a very different behaviour between the two metrics. This behaviour differs from the other two selection operators and is reflected in the negative correlation value shown in Table 5.6.

Figure 5.5 shows the PBIL results for the comparison between the local information gain and inter-centroid angle. Here, as reflected in Table 5.5, we see a clearer negative correlation between the two metrics in all selection methods. None of the selection methods appears to show a distinctly different behaviour from the others, unlike the GA, and none shows the peak that is also found in the GA results for either metric.

As noted in [2], this difference in behaviour may be due to the PBIL evolving each population using a probabilistic model. This tends to be a gradual process, causing local information gain to accumulate before it is observed in the inter-cluster angle values. This is also true when looking at the results of the global information gain when compared to the angle from origin. Here, the PBIL reaches the maximum information gain value much later than the GA with an almost linear ascent. The GA shows a much steeper global information gain, reaching the maximum within the first 20 generations. These differences in behaviour help show that the metrics being used are capable of detecting the differences in algorithm behaviour on the same optimisation problems and selection methods.

5.4.4 Entropy and Angular Correlation

Here, in Table 5.5 we show the results of performing Spearman Rank correlation between the two sets of metrics. This is achieved by calculating the mean angular metric and information gain, per generation, for all 100 optimisation runs. The results are broken down by algorithm, optimisation problem and selection method. Spearman correlation was selected for this analysis for two specific reasons. First, the KLd metric is non-linear, as seen in its definition in Chapter 4, therefore we cannot assume a linear relationship between the KLd and angular-based metrics developed for this thesis. Secondly, we also cannot assume that the results are normally distributed. Due to these two conditions, the Spearman correlation was used as opposed to the popular alternative of Pearson's. Values in bold in the table are results that could not be considered statistically significant as the associated p-value was higher than the threshold of 0.05.

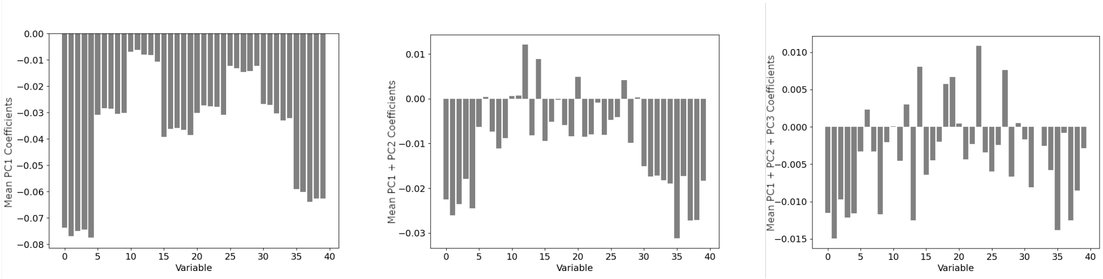
The results show that the "Global to Origin" comparison, between the global information gain and angle to origin metrics, shows a markedly higher level of correlation than the local values. Here, across all algorithm-problem-selection combinations, the lowest value attained was 0.853 for the GA on Trap 5 using Truncation 20%. Interestingly, the PBIL also shows a higher correlation between the two than the GA in general and all results that did not pass the p-value test are found in the GA section. Overall, this table highlights the fact that, especially in the Global to Origin results, the decomposition and generalisation of the solution populations to a single vector were able to retain geometrically important features. When compared to the KLd metric which used the full population data, significant overlap between the two methods can be seen. As PCA is a non-destructive, linear process, the structures or relationships detected via our geometric-based method must exist in the original data as PCA is not introducing new information to the dataset.

Algorithm	Problem	Selection	Global to Origin	P-Value	Local to Inter	P-Value
GA	1D Checker	Truncation 20%	**0.990**	**5.19E-01**	0.519	1.11E-04
GA	1D Checker	Truncation 50%	0.999	5.76E-68	-0.422	2.27E-03
GA	1D Checker	Tournament	0.990	5.15E-43	0.504	1.92E-04
GA	Royal Road	Truncation 20%	0.977	5.05E-34	0.769	6.95E-11
GA	Royal Road	Truncation 50%	0.987	5.94E-40	**0.229**	**1.09E-01**
GA	Royal Road	Tournament	0.974	1.03E-32	0.818	4.30E-13
GA	Trap-5	Truncation 20%	0.853	3.62E-15	0.554	2.99E-05
GA	Trap-5	Truncation 50%	1.000	2.59E-105	**0.216**	**1.31E-01**
GA	Trap-5	Tournament	0.905	1.83E-19	0.722	3.15E-09
PBIL	1D Checker	Truncation 20%	1.000	0.00E+00	-0.858	1.78E-15
PBIL	1D Checker	Truncation 50%	1.000	1.22E-83	-0.796	5.07E-12
PBIL	1D Checker	Tournament	1.000	0.00E+00	-0.754	2.69E-10
PBIL	Royal Road	Truncation 20%	1.000	0.00E+00	-0.828	1.20E-13
PBIL	Royal Road	Truncation 50%	1.000	0.00E+00	-0.790	8.76E-12
PBIL	Royal Road	Tournament	1.000	0.00E+00	-0.814	6.62E-13
PBIL	Trap-5	Truncation 20%	0.999	7.21E-67	-0.728	2.12E-09
PBIL	Trap-5	Truncation 50%	1.000	4.32E-74	-0.782	1.93E-11
PBIL	Trap-5	Tournament	0.970	4.08E-31	-0.717	4.85E-09

Table 5.6 Spearman Correlation Coefficient Between Information Gain to Angular-Based Metric. Bold values indicate where the p-value associated with the correlation was > 0.05, failing to reject the null hypothesis

5.4.5 Principal Component Coefficient Values

The mean coefficients calculated for each algorithm and problem combination can be seen in Figures 5.7 to 5.12. Here, the mean component coefficients are calculated across the first principal component and all runs. As seen in Table 5.5, the first component explains a significantly larger level of variance in the data than the others. The effects of including components 2 and 3 in the calculation can be seen in Figure 5.6. Here, we show the GA results on the Trap 5 problem using Truncation 20% selection as an example. The effect of using components with considerably lower levels of explained variance than the first can be seen across these combinations. It is suspected that the addition of these components has introduced some level of noise into the dataset, making the detected problem structure more difficult to identify. To this end, the analysis is performed using only the first component which has a significantly higher level of explained variance than the others.



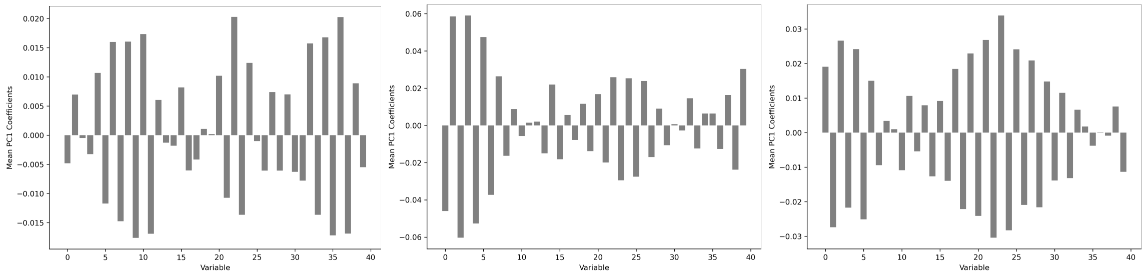
(a) Components - [1] (b) Components - [1,2] (c) Components - [1,2,3]

Fig. 5.6 Mean Principle Component Coefficients Over First 3 Components - GA, Trap5, Truncation 20%

The 1D Checkerboard problem results, presented in Figures 5.7 and 5.8, demonstrate that the GA discovered solutions where adjacent variables have opposing values, a pattern also observed in PBIL's results. This reflects the fitness function's structure, where the two global optimal solutions exist – each being a mirror of the other. However, both algorithms occasionally deviate from this expected alternating pattern with this effect more noticeable in the PBIL's Truncation-50 results (Figure 5.8.b).

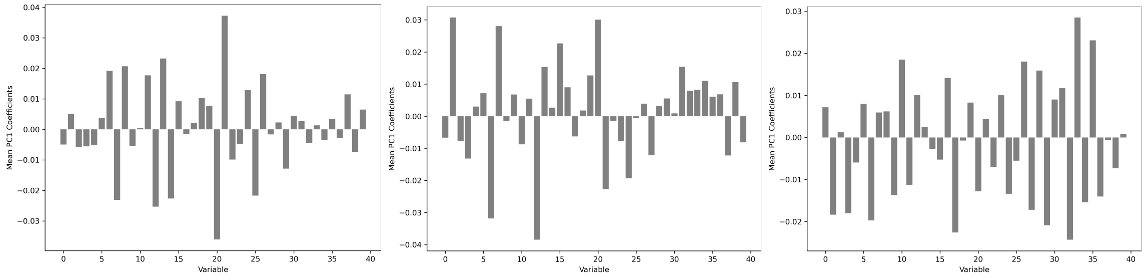
For the Royal Road problem, shown in Figures 5.9 (GA) and 5.10 (PBIL), a similar pattern to the Trap 5 can be seen in the results. The GA shows partial recognition of the problem structure, with some blocks of 5 bits displaying similar values distinct from other adjacent blocks. The PBIL's results do not share this pattern or any pattern that aligns with the fitness function's structure.

For the Trap5 problem, the GA's results are shown in Figure 5.11 and PBIL's in Figure 5.12. The GA results across all three selection methods show uniformity in all 8 blocks of 5



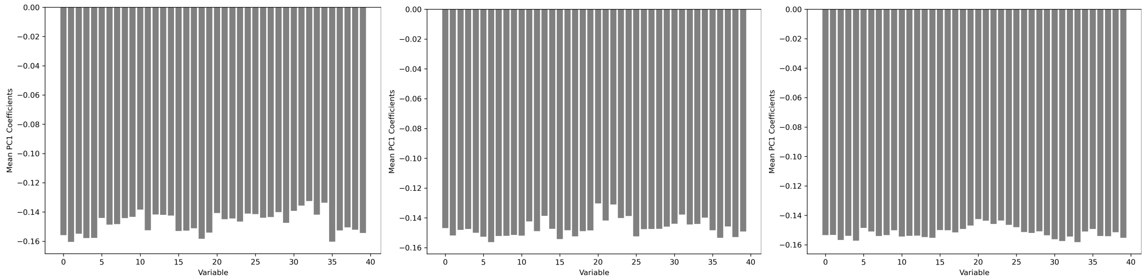
(a) Truncation 20% (b) Truncation 50% (c) Tournament

Fig. 5.7 GA Mean Coefficient Values on 1D Checkerboard by Selection



(a) Truncation 20% (b) Truncation 50% (c) Tournament

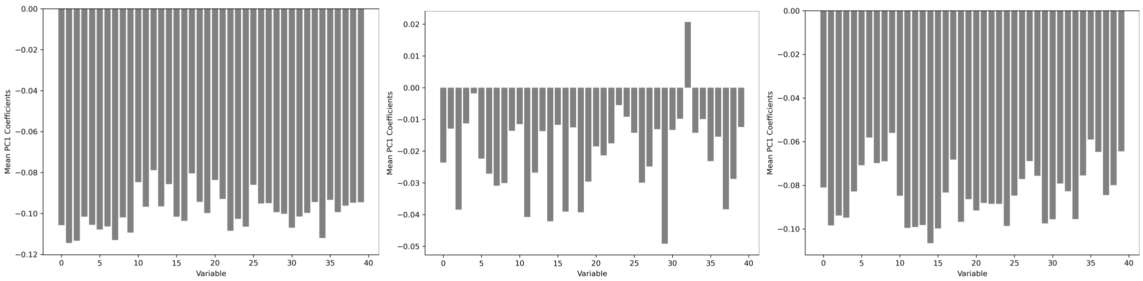
Fig. 5.8 PBIL Mean Coefficient Values on 1D Checkerboard by Selection



(a) Truncation 20% (b) Truncation 50% (c) Tournament

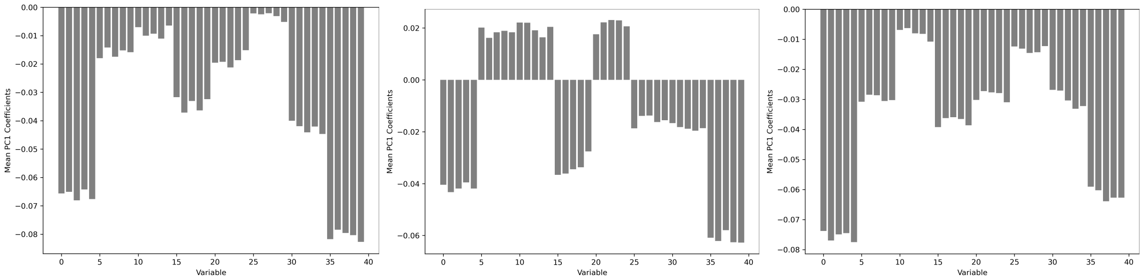
Fig. 5.9 GA Mean Coefficient Values on Royal Road by Selection

consecutive bits, aligning with the expected fitness function structure of Trap5, where each 5-bit block needs to be identical to achieve maximum fitness. In contrast, PBIL's results do not reflect this correlation with the expected fitness structure, likely due to PBIL's inability to detect multivariate bit interactions.



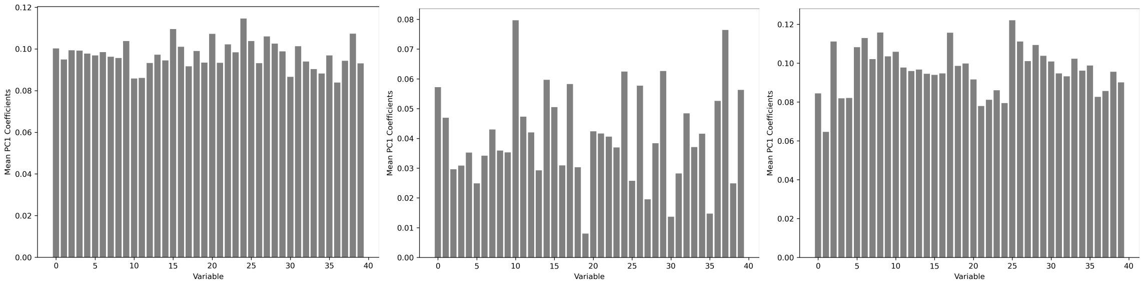
(a) Truncation 20% (b) Truncation 50% (c) Tournament

Fig. 5.10 PBIL Mean Coefficient Values on Royal Road by Selection



(a) Truncation 20% (b) Truncation 50% (c) Tournament

Fig. 5.11 GA Mean Coefficient Values on Trap5 by Selection



(a) Truncation 20% (b) Truncation 50% (c) Tournament

Fig. 5.12 PBIL Mean Coefficient Values on Trap5 by Selection

5.5 Summary

The results shown in this chapter provide key insights into the ability to detect geometric features from the search trajectories of EAs post-dimension reduction and how they relate to the fitness function of an optimisation problem. The results show that there is a strong correlation between the angle from origin results to that of global information gain, across both algorithms and all three selection methods, representing differing levels of selection pressure. This correlation is, however, lower in strength when comparing the inter-cluster angular results to that of local information gain. These results show that it is possible to detect differing algorithm search behaviours on the same problems between the GA and PBIL in the inter-cluster angle values. By using both metrics, we show that the KLd results validate those for the inter-centroid angle measurements when observing the "Global" change in values. While both metrics use the same sets of solutions, they are fundamentally different in that the our novel methods utilise vector similarities and KLd is based on differing probability distributions. The benefit to both User and Developer in using both is that, firstly, the KLd informs them of the changing dynamic between generations in terms of solution distribution. Secondly, the angular-based metrics inform them of the changing algorithm position and how this relates to population diversity. By using both we can present how changes in one metric may or may not be observed in the other. The inter-centroid measurements also provides the Developer the ability to compare two EAs with similar search methodologies on the same problem. By comparing the Local and Global results, it may be possible to distinguish between EAs and use this knowledge to attribute differences in overall performance to behaviours observed using these metrics.

The GA has shown higher sensitivity to selection pressure when evolving populations towards higher quality solutions, as seen in the global information gain values. This is less prominent in the PBIL results which show that selection pressure has had a lower impact, both in local and global metrics. It should be noted that this difference is also seen in its overall performance which is due to its univariate nature. These results are important in that they highlight that, even when reduced to only three dimensions and represented as a single vector, there is sufficient remaining structure in the EA trajectories for the angular-based metrics to detect a significant level of algorithm behaviours when compared to the entropy based metrics.

The PCA decomposition method results in the creation of a variance-based subspace in which the components identify the direction of maximum variance in the data. The coefficients of these components identify the contribution each of the original variables has in the definition of those components. By observing the mean coefficient values across multiple runs, the results can indicate any consistent relationship between those variables and components. It has been shown in our analysis that these coefficients can be used to represent the learnings of each algorithm in the context of variable contributions to overall solution fitness.

As PCA is designed to identify the direction of maximum variance, the detection of any problem structure in the analysis is dependent on the level of contribution such structure has on the overall variance of the data, captured by the populations of solutions. This suggests that higher impact problem structures can be detected in the population provided they have a high impact on solution quality – blocks of bits or adjacent bits that significantly contribute to the variance in the data are highlighted by the decomposition process. This can be seen in the GA results of the mean coefficient values for both the 1D checkerboard and Trap 5 problems, where the values reflect the fitness function structure of the optimisation problems. The PBIL shows a lower level of structure captured when compared to the GA, again most likely due to its univariate nature, meaning higher-level features are less likely to be captured.

Chapter 6

Trajectory Mining of Variable Contribution and Alignment to Fitness.

6.1 Introduction

In Chapter 5 we have shown how the application of PCA to the search trajectories of two simplistic metaheuristics – a GA and univariate EDA – can be used to extract explanatory features to relate search behaviour to detected low-order variable interactions key to finding high-quality solutions. In this chapter we extend the application of PCA to EA search trajectories by applying two methods of variable importance detection to the search trajectories generated by a collection of EAs. These methods, defined in Chapter 4, are a measure of mean variable contribution and of variable alignment to fitness change. The first metric aims to detect the mean variable influence across multiple optimisation runs in relation to a specific component. The second measures the proportional variable alignment to fitness. This metric is designed to identify which component, at any stage or window of generations of a search trajectory, is dominant in terms of contribution to fitness gains during that period. To further extend the application of PCA to EA search trajectories, we use real-valued vector solution representations generated by solving a set of real-value optimisation problems.

By measuring the dominance of the components, we can relate geometrically sensitive features derived from trajectories to algorithm search behaviour. This aim is analogous to the concept of Causality in machine learning, in which, as noted in [127], while causality and inference of variable relationships require background knowledge in the problem, explainable ML models that detect such interactions can be used to support existing findings "...to provide a first intuition of possible causal relationships within the available data...". By simplifying the explanation representation into sets of variable relationships, we can provide more interpretable and concise explanations of the algorithm's behaviour, aiding in the understanding and trustworthiness of the AI system [240].

The main contributions of the work in this chapter relate to the answering of research question (Q4) by furthering progress towards objective (O3) and (O5). To achieve this, we expand on the set of explanatory features we aim to generate from EA search trajectories. Here, we show that trajectory mining techniques can be applied to real-valued problems and are capable of creating features that aid in explaining algorithm search behaviours and how they differ on the same problem to other EAs. These features are also capable informing users regarding solution quality drivers at key stages of the search. The results and experimentation in this chapter are an extension of techniques published in [3].

6.2 Experimental Setup

6.2.1 Optimisation Problems

To generate the data required for the analysis in this chapter, the noiseless variant of the Black Box Optimisation Benchmarking (BBOB) [222] problem set was used. This problem set, as outlined in Chapter 3, is a well-known and often used problem set for the measurement and comparison of optimisation techniques. The collection contains 24 real-valued problems with a range of properties including low or moderate conditioning, unimodal, multimodal and both strong and weak global structure examples. An important aspect of this problem set is that, while called "Black Box", all problems in the set have known optima and function values. The implementation of these problems was taken from from BBOBtorch Python library of the same name [241]. This provided a flexible framework that allowed for rapid implementation of the problem set. Each BBOB function represents an instance of a problem, and for our study, all algorithm runs were executed on the same instance of each problem to maintain consistency.

The set of population-based metaheuristics chosen was that of a Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm, a Differential Evolution algorithm (DE), a Genetic Algorithm (GA) and a Particle Swarm optimisation algorithm (PSO). The mechanisms of these are outlined in detail in Chapter 2. Each of the BBOB problems was solved a total of one hundred times by each of the algorithms and the same problem instances were used throughout. Each optimisation run was initialised with a randomised starting population - this allowed for a direct comparison of search behaviours on the same problem landscape. Shown in Table 6.1 are the collection of 24 BBOB problems and their defining features, as defined in [222].

Function	Problem	Problem Features
F1	Sphere	Seperable
F2	Ellipsoidal	Seperable
F3	Rastrigin	Seperable
F4	Büche-Rastrigin	Seperable
F5	Linear Slope	Seperable
F6	Attractive Sector	Low or Moderate Conditioning
F7	Step Ellipsoidal	Low or Moderate Conditioning
F8	Rosenbroch	Low or Moderate Conditioning
F9	Rosenbroch Rotated	Low or Moderate Conditioning
F10	Ellipsoidal	Unimodal High Conditioning
F11	Discuss	Unimodal High Conditioning
F12	Bent Cigar	Unimodal High Conditioning
F13	Sharp Ridge	Unimodal High Conditioning
F14	Different Powers	Unimodal High Conditioning
F15	Rastrigin	Multi-Modal Adequate Global Structure
F16	Weierstrass	Multi-Modal Adequate Global Structure
F17	Schaffers F7	Multi-Modal Adequate Global Structure
F18	Schaffers F7 Ill-Conditioned	Multi-Modal Adequate Global Structure
F19	Composite Griewank-Rosenbroch	Multi-Modal Adequate Global Structure
F20	Schwefel	Multi-Modal Weak Global Structure
F21	Gallaghers Gaussian 101-me Peaks	Multi-Modal Weak Global Structure
F22	Gallaghers Gaussian 21-hi Peaks	Multi-Modal Weak Global Structure
F23	Katsuura	Multi-Modal Weak Global Structure
F24	Lunacek bi-Rastrigin	Multi-Modal Weak Global Structure

Table 6.1 BBOB Function Properties

To illustrate the results of our analysis, a subset of four BBOB problems was selected. While these four have been selected to illustrate our methodologies, Appendix B contains the results for all 24 BBOB problems which are noted in each of the relevant areas of the Results section of this chapter. This selection was comprised of one function from each main group of problem types. The first of the four selected was the Rastrigin Function (F3). This provided an example of a separable function from the collection. The second selected was the Bent Cigar Function (F12) to provide an example of a function with high conditioning while remaining uni-modal. The third selected is a modification of the Rastrigin

Function (F15) as an example of a multi-model function with an “adequate” global structure as described in the functions implementation documentation. This modification alters the regularity and symmetry of the original function. Lastly, the Katsuura Function (F23) was selected to provide trajectories for a multi-model function with “weak” global structure as again described in the functions implementation documentation. This last function was also selected as an example of a considerably more rugged landscape with a large number of global optima [242]. This selection provided a representative sample of the functions available in the BBOB collection.

6.2.2 Algorithm Run Settings

The population-based metaheuristics used in this chapter were selected as they provide a significant range of search techniques while remaining approachable enough to relate detected features to their designed search behaviour. The algorithm implementations used were created and packaged as part of the PYMOO [43] Python package.

The settings used for our optimisation runs, detailed in Table 6.2, included the number of runs, maximum generation count, population size, and problem dimensionality. We used 100 runs per algorithm-problem pair with 300 generations, ensuring ample data for analysis. A fixed population size was chosen to standardize the trajectory structures and ensure consistent data points across all algorithm and problem pairs for the decomposition process. While adaptive population size adjustments, as seen in [243], can enhance performance, it may not be advantageous for 10-dimensional problems. We settled on a population size of 50, considering the diversity of algorithms. As shown in [244], PSOs can benefit from larger-than-traditional population sizes. Thus, we opted for the higher end of their "classical" range, maintaining consistency across all algorithms with the value of 50 which provides a large enough selection pool for all algorithms used.

Runs	Max. Gens	Pop. Size	Dimensions
100	300	50	10

Table 6.2 BBOB Shared Algorithm Run Settings

For each algorithm, we use a set of algorithm-specific runtime parameter settings as recommended in the implementation documentation. These parameters cover algorithm-specific settings such as crossover rate, velocity and specific DE variants and can be found in Table 6.3. It is important to highlight that this study did not focus on optimising the algorithm’s performance. Our primary concern was generating higher-quality solutions to facilitate our analysis. Consequently, whenever possible, we adhered to the default values and operators provided in the PYMOO documentation. This approach was adopted because the main focus of this chapter is not the comparison of algorithm performances as is common when solving the BBOB set. The default settings outlined the PYMOO documentation, and their framework was used to generate the required search trajectories. This allowed us to test our methods on identifying differing algorithm behaviour under default conditions, not the like-for-like performance comparisons typically seen from these problems.

Algorithm	σ	Restarts	Max. F-Evals
CMA-ES	0.1	0	∞

Algorithm	Variant	CR	F
DE	DE/rand/1/bin	0.9	0.5

Algorithm	Selection	Mut.	eta	Crossover
GA	Tournament	Polynomial	3	1-Point

Algorithm	ω	c_1	c_2	Adaptive	Init. Vel	Max. Vel Rate
PSO	0.9	2	2	TRUE	Random	0.2

Table 6.3 BBOB Algorithm-Specific Run Settings

Covariance Matrix Adaptation Evolution Strategy

The only parameters specifically set by ourselves for the CMA-ES algorithm was the σ value which controls the initial standard deviation in each coordinate and the generation count stopping criteria. These values were set to $\sigma = 0.1$ and 300 respectively.

Due to the specific implementation of CMA-ES used in this chapter, its trajectories needed post-processing. We set the termination criteria of the algorithm to the maximum number of generations used and the maximum fitness evaluations to ∞, however, we found that CMA-ES often converged and the run was terminated regardless of our settings. While it would have been possible to avoid this behaviour by altering additional termination criteria found in this implementation or using a restart strategy, we decided to extend the trajectories to conform to the structure of the other algorithms. This was done to avoid the possibility of inadvertently negatively impacting the CMA-ES' behaviour by altering key internal settings to force a longer search process. To standardize, CMA-ES trajectories were artificially extended to match other algorithms by replicating the final solution set. While normalizing trajectory lengths was considered, it was avoided to prevent potential data distortion and misrepresentation of fitness changes or features in the trajectory that are sensitive to geometric locations in the search path.

Differential Evolution

In this chapter we use the "DE/rand/1/bin" variant of the DE algorithm. This means that the base vectors are randomly chosen, one vector difference is used to mutate the population and binomial distribution is used. The crossover rate (**CR**) for the DE was set to 0.9. This value was selected as it was a recommended value from the documentation. It was noted that a lower CR value tends to help with optimising separable functions however a default of 0.9 for use in a broader set of functions was used. The F value, which represents a weighting or scaling factor shown in Chapter 2, was set to 0.5 as this was the default.

Genetic Algorithm

The implementation of a Genetic Algorithm used was a $(\mu + \lambda)$ algorithm, where μ is an initial population size and λ is the size of the population of offspring solutions once the internal operators have been used. This algorithm generates each successive population of solutions by performing the Selection, Crossover and Mutation operations on the parent population, details of which are shown in Table 6.3.

Here, tournament selection was used to select two parent solutions randomly selected from the population. A single-point crossover operator was used to handle crossover. The mutation used was a polynomial [238] function, details of which can be found in [239], and the setting eta was set to 3.0. The higher the value of eta the more similar and less mutated the child solution will be.

Particle Swarm Optimisation

Implementation-specific run settings for the PSO include setting the ω value to 0.9, c_1 and c_2 set to 2.0. The Adaptive setting was set to true, specifying that ω, c_1, and c_2 are changed dynamically over time based on the spread from the global optimum. The initial velocity of each particle was set to a random value between 0 and 1 and the maximum velocity rate was set to 0.2. As with the other algorithms, these were selected as they were the default or recommended values as per the implementation documentation.

6.3 Analysis Methods

6.3.1 Mean Variable Contributions

The mean variable contribution metric, seen in Chapter 4, is an extension of the methods used in [6], where we measure the variable contribution across a given component over multiple runs rather than the values. This is done as the principal component coefficient values were used in Chapter 5 to highlight the strength and direction of relationships between variables across a given component. We now build on this approach by using the variable contributions which help show the importance of each variable in explaining the observed variance within the data, across a given component. In dimension reduction, this process is often used to determine which variables should be selected when performing feature selection. This difference in approach allows us to understand and convey the importance of each variable in its ability to preserve key information during the decomposition process. By observing this at different stages of the search, we can infer which variables contributed more than others to the direction of search at that time, along the fitness gradient, and as such contributed more to the achievement of higher quality solutions. The method of contribution calculation is repeated in Equation (6.1) for reference.

$$C_i' = \sum_{k=1}^{N} \left(\frac{\left(\sum_{j=1}^{n} p_j^i x_j^k \right)}{N} \right) p^i. \tag{6.1}$$

6.3.2 Proportional Variable Alignment to Changing Fitness

The equations necessary to calculate a variable's alignment to changing fitness, or the fitness gradient of an optimisation run, are repeated in Equations (6.2) to (6.4). Here, our definitions of how the proportion of variables in a given generation that are aligned are calculated.

$$V_j^i(x) = \left\{ \begin{matrix} 1 \text{ if} & x \cdot p^i p_j^i \geq 0 \\ -1 & \text{Otherwise} \end{matrix} \right\} \quad (6.2) \quad G_i(x_j) = \left\{ \begin{matrix} 1 \text{ if} & \left(V_j^i(x) \geq 0 \text{ and } x_j \geq \bar{T}_j \right) \\ 1 \text{ if} & \left(V_j^i(x) \leq 0 \text{ and } x_j \leq \bar{T}_j \right) \\ 0 & \text{Otherwise} \end{matrix} \right\} \tag{6.3}$$

$$PG_i = \frac{\sum_{k=1}^{N} G_i(x^k)}{N} \tag{6.4}$$

Here, $V_j^i(x)$ determines the sign of each variable a the given solution x, using the score value calculated as $(x \cdot p^i)$ across component p_i. $G_i(x_j)$ (Equation (6.3) returns a binary value depending on whether the variables sign calculated in (6.2) is reflected in its relative position to its original data mean value. Lastly, the proportion of variables that are aligned is calculated as in Equation (6.4) and represented as PG_i.

6.3.3 Fitness Quartiles

The methodology for determining the generation number at which a specific fitness threshold is reached within an optimisation run is shown here. The fitness threshold achievement is used as a measure for understanding the progress of an optimisation process. The process of calculating the generation number at which this occurs can be seen in Chapter 3 and the equations are repeated here for reference, seen in Equations (6.5) to (6.7).

$$f_g^* = \min_{x \in X_g} F(x_g) \tag{6.5}$$

$$\Delta f^* = f_0^* - f_l^* \tag{6.6}$$

$$g_q = \min\{g : f_0^* - f_g^* \geq q\Delta f^*\} \tag{6.7}$$

As noted in the definition in Chapter 3, we use $F(X_g)$ to represent the collection of all fitness values in a given generation g and f_g^* contains the fitness value of the best-found solution in g. The fitness of the best-found solution in the initial starting population of solutions, that will be used as a comparison point, is shown as f_0^*. To calculate the total fitness change from the initial solution to the final, best-found solution (f_l^*) in a run we use Δf^* to represent the difference.

This allows us to formalise the method of fitness quartile calculation as shown in Equation (6.7). Each quartile represents approximately 25% of the total fitness change observed in an optimisation run, with thresholds used in this chapter being 25%, 50%, 75% and 99%. To calculate these values for any given optimisation run, we use the values [0.25, 0.5, 0.75, 0.99] as q, allowing us to calculate the minimum generation at which a given fitness threshold is achieved during that run, stored in g_q. This is then used to partition the optimisation runs into quartiles for later analysis. This process is used in conjunction with the mean variable contribution and proportional variable alignment methods to allow both methods to be applied to subsets of the search trajectory, defined as the generations of solutions contained in the quartiles. To achieve this, the mean generation at which each quartile is achieved is calculated across all 100 runs. Those generations are then used to define the windows used in the analysis for all subsequent tests.

6.4 Results

6.4.1 Algorithm Performance

The results of the optimisation runs are summarised in Table 6.4. Shown in this table are the algorithms used and the problems solved. The table also shows F-Opt, which is the optimal value for all instances of a given problem. The median best fitness represents the median value of the best fitness achieved across 100 independent runs for each algorithm-problem combination. Lower values indicate better performance. The Min and Max values are the minimum and maximum values found in the final generation of solutions, showing the range of solution qualities found in the converged populations across all runs. We also show the standard deviation of that collection and the distance to the optimal - this is a measure of the distance, in terms of fitness value, between the median best fitness and the problem's optimal value. Highlighted in bold is the best-performing algorithm for each problem, measured by the distance to the optimal. Table B.1 in Appendix B contains the optimisation results for all 24 BBOB problems.

Alg	Prob	F-Opt	Median Best F.	Min	Max	Std	Dist. To. Opt
CMAES	F3	129.88	137.7932	130.8724	152.6234	3.475602	7.913182
DE	F3	129.88	153.4348	145.2051	162.9933	4.142318	23.55484
GA	F3	129.88	129.8947	129.8846	129.9456	0.012071	0.014684
PSO	F3	129.88	132.9703	129.8837	145.644	2.896078	3.090268
CMAES	F12	1000	1000	1000	1006.39	1.282499	0.000183
DE	F12	1000	1000.514	1000.093	1005.503	1.133905	0.51358
GA	F12	1000	1016.717	1001.443	1261.902	28.04532	16.71741
PSO	F12	1000	1001.153	1000	1010.331	2.529329	1.152771
CMAES	F15	1000	1006.921	1000	1024.932	4.739127	6.920837
DE	F15	1000	1035.727	1022.694	1046.386	5.215436	35.72729
GA	F15	1000	1025.017	1006.021	1055.375	10.87465	25.01678
PSO	F15	1000	1020.762	1004.943	1064.264	11.90002	20.76242
CMAES	F23	129.88	129.9195	129.8837	131.6227	0.194818	0.039548
DE	F23	129.88	131.4158	130.8393	132.0701	0.264654	1.535779
GA	F23	129.88	130.8568	130.2254	131.9019	0.370717	0.976804
PSO	F23	129.88	131.4048	130.4849	132.1611	0.359095	1.524785

Table 6.4 BBOB Algorithm Run Results F3, F12, F15 & F23

The results show that the CMA-ES is highlighted as being the overall best-performing algorithm in 3 of the 4 problems by consistently achieving solutions with minimal deviation from the optimal values except for problem F3 in which the GA is the best performer. The Algorithm Run Results shown in Table 6.4 demonstrate that all algorithms are capable of discovering high-quality solutions throughout their optimisation runs. As such, the trajectories generated by each algorithm are suitable for further analysis.

The results show that, for the F3 problem, the GA stands out as having the best performance. It achieved the lowest mean fitness distance to the optima with a value of 0.01. This indicates that it consistently approaches the optimal fitness value of the F3 function. In relation to the other algorithms, the GA also has the lowest standard deviation. Both CMA-ES and the PSO results show final fitness values below 8 and the DE results show the poorest performance with a value of 23.55. The low standard deviation and range between minimum and maximum fitness found also highlight the GA's better performance.

In problem F12, DE shows a slightly lower standard deviation of 1.13 showing that, while it did not consistently find the optimal as the CMA-ES did, it has a lower spread of fitness values however its mean distance to the optimal solution was 0.514. Overall, The CMA-ES outperformed the other algorithms on this problem with a near-zero value for the distance to the optimal solution as compared to the PSO (1.15) and the GA (16.71) which shows the worst overall performance on this problem. It should be noted however that both the CMA-ES and PSO had the optimal solution in their final generations of solutions as seen in the Min. fitness found. This was closely followed by the DE which is reflected in the median best fitness values.

The results for problem F15 indicate that the CMA-ES performed best, with the lowest distance to the optimal, standard deviation and had found the optimal solution of 1000 - the only algorithm to do so in the final generation of solutions as seen in the Min. fitness value results. This however was not the norm as the mean distance to the optimal was 6.92. Both GA and PSO had a higher mean distance to the optimal value (25.02 and 20.76, respectively) and the DE had the highest with a value of 35.73. Interestingly the GA and PSO have the highest standard deviations with values of 10.87 and 11.90, potentially suggesting a wider search of the space.

For the F23 results, we see that all algorithms have relatively good performance, as seen in the lower distance to optimal (129.88) and standard deviation values. The CMA-ES outperformed the other algorithms with the lowest mean distance to the optimal with a value of 0.04, with the next closest being the GA with 0.98.

6.4.2 Component Explained Variance

The components generated from the decomposition of the trajectories are ordered by the magnitude of their eigenvalues. By dividing the eigenvalue of each component by the sum of the eigenvalues we can show the percentage of variance each PC can explain in the data. Shown in Figure 6.1 are the explained variance ratio results across all 10 components for each pair of algorithms and problems. The values shown in the figure are the mean explained variation, in percentage, across all 100 runs for each algorithm-problem pair.

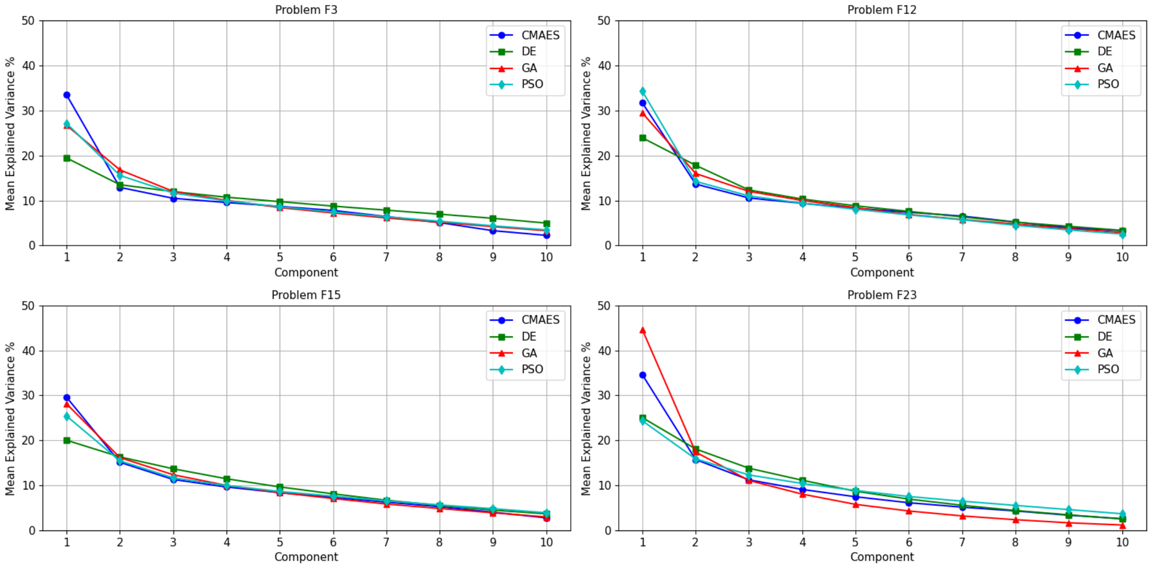


Fig. 6.1 Algorithm Explained Variance Ratio

For all algorithm-problem pairs, we show that there is a diminishing level of explained variance with each successive component which is the expected behaviour. We see that the DE results show a lower mean explained variation percentage than the other algorithms in problems F3, F12 and F15 with problem F23 being the exception - in F23 the DE results more closely match those of the PSO across all components.

In functions F3, F12 and F15 the level of explained variance in the first component is between 20% and 35% for all algorithm-problem pairs however this is not the case for F23. In this problem, we see that the GA has a significantly higher value of 45%, 10% higher than the next nearest algorithm, CMA-ES, across the first component. This difference however is short-lived as by the second component, all algorithms have between 16% and 19%. In F3, we see that the CMA-ES has a higher level of explained variance than the other algorithms however this quickly drops in the subsequent components, as do the results of the other algorithms.

When comparing these results to the algorithm performances shown in Table 6.4 we see that there does not appear to be a clear pattern between the level of explained variance in the dominant component (component 1) and the overall algorithm performance. In function F3 we see that the CMA-ES has a higher mean level of explained variance however the best-performing algorithm was the GA across all measures. In F12 and F15 the CMA-ES, PSO and GA results are closely matched across all components however the CMA-ES algorithm outperforms all others in terms of mean proximity to the optimal solution. In Function F23 we see that the GA has a significantly higher mean explained variance in the first component however again the CMA-ES outperforms all other algorithms in all measures.

The results do show however that there are subtle differences in the level of structure captured across each component between algorithms and problems, implying that their different search behaviours have some effect on the level of explained variance as calculated by the PCA decomposition. The results also show that there are consistently higher levels of variance explained by the first component across all results. This indicates that there are dominant features or key structures in the search trajectories that are being captured by our approach, otherwise, a more evenly distributed level of variation would be attributed to all components.

6.4.3 Fitness Quartile Windows

To allow for a fair comparison of variable contribution changes over the course of a search trajectory, we calculated the mean generation number in which the algorithm had achieved a set of fitness goals. This value was then rounded to the nearest whole number for sub-setting our data by generation. The goals were set at 25%, 50%, 75% and 99% of total fitness reduction in the set of minimisation problems. This was calculated for each Algorithm-Problem pair and presented in Table 6.5. These values are the mean value across all 100 runs of each pair and are calculated using Equations (6.5) to (6.7). Each table shows the upper generation that defines each of the windows. These values were used to subset the trajectories into windows based on generation number. The corresponding fitness windows for all 24 BBOB problems can be found in Appendix B.2.

As previously mentioned, the CMA-ES trajectories were extended to the same length as the other three algorithms through the repetition of the final population. This allowed for the comparison and plotting of the search paths on a generation-by-generation basis across the entire search trajectory. This method was used to generate the plots showing the results of the variable alignment proportions shown in Section 6.4.5. The mean variable contribution calculations may be sensitive to this extension as it would affect the mean value calculated in the final window if that target was set to 100% of fitness. By using the 99% target the

Alg	Prob	25% Window	50% Window	75% Window	99% Window
CMAES	F3	4	11	27	65
DE	F3	4	9	21	209
GA	F3	3	6	12	70
PSO	F3	3	7	25	194
CMAES	F12	2	4	9	26
DE	F12	4	6	11	39
GA	F12	2	3	6	21
PSO	F12	1	2	3	13
CMAES	F15	4	11	27	67
DE	F15	4	7	21	185
GA	F15	2	6	14	85
PSO	F15	2	6	26	95
CMAES	F23	13	40	79	119
DE	F23	21	43	83	157
GA	F23	11	23	50	180
PSO	F23	21	39	88	186

Table 6.5 Fitness Window Generations by Algorithm-Problem Pair

final window can be considered close to the point of convergence with less chance of being affected by a large number of low-diversity solutions at the end of the trajectory.

These results show that the generation that defines each fitness quartile varies significantly between problems for each algorithm. In problem F3 we see that the 25% windows are achieved within 4 generations across all algorithms. This is in contrast to problem F23, which has a significantly more rugged landscape, in which the same milestone is achieved within 11 (CMA-ES) to 21 (PSO, DE) generations.

In problem F3 we see that CMA-ES and the GA can achieve each of the fitness milestones earlier than the DE or PSO, especially when achieving the final 99%. The DE and PSO both take significantly longer to achieve this window, with 209 and 194 generations, on average, taken before this is done. Both the CMA-ES and GA take between 65 and 70 generations to achieve the 99% window of fitness change.

The results for problem F12, in comparison to those of F3, show a closer grouping of generations between the algorithms. These results show that higher-quality solutions are quickly found by all algorithms, with the PSO within a single generation finding considerably better-quality solutions. All algorithms achieved the 99% window within 39 generations, with the PSO achieving this within only 13 and the DE taking the longest at 39.

Problem F15 shows that most algorithms perform broadly similarly in terms of fitness reduction with the exception of the DE algorithm. It maintains a similar level of reduction as the others however the final window takes considerably longer than the others with 185 generations. The next algorithm to this is the PSO, requiring only 95.

Lastly, the results for function F23 show that there were no significantly different results between the algorithms in the first three quartiles. The 99% window does show a greater difference between algorithms, with the CMA-ES taking 119 compared to the PSO's 186 generations, however, the GA takes only 6 less at 180 generations.

6.4.4 Mean Variable Contributions

Using the fitness windows from each algorithm-problem pair, the mean variable contribution (C_i') was calculated across the first three PCs. To achieve this, the generation bounds identified for each fitness quartile were utilized to partition the individual optimisation runs into four distinct sections. Then, contribution values for each variable were computed within each window, and the means across all 100 runs were determined for each quartile. The Mean Variable Contribution results are plotted in Figures 6.2 - 6.17. These are the mean contribution values for each variable across the first three PCs' windows. These values illustrate a variable's relative influence on the overall position within a PC, compared to other variables. By averaging across 100 runs and fitness windows, we can highlight variables' significance in determining algorithm positions during critical trajectory phases. Contribution values aid in identifying key variables driving positional shifts towards improved fitness solutions as algorithms evolve their populations. The ability to explain which key variables are driving this positional shift to the Developer has the potential to aid in algorithm comparison and fine-tuning. By observing, at each stage of the search, which variables are driving the shift the Developer can assess whether one algorithm is capturing the same knowledge as others in terms of variable importance. This can in turn be used to compare overall search performance and differences between detected importance levels across multiple EAs on the same problem. The results in Figures 6.2 - 6.17 can be used to highlight these differences to the Developer and can also be used to measure, between design iterations, any change in EA search behaviour.

Function F3

The results for Function F3 are shown in Figures 6.2 to 6.5. Across PC1, The CMA-ES, DE and GA algorithms display the same subset of variables with higher contributions during the 25% window, specifically variables [2, 3, 4, 8]. The PSO has a highly similar subset of [3, 4, 5, 8]. This subset remains consistent across all four fitness windows in PC1, with PSO results indicating a slightly higher value for variable 4 compared to variable 2. This highlights these specific variables as of a higher influence in directing the search towards better quality solutions across all algorithms. As fitness decreases, the overall variable contribution across all PCs tends to decline. Notably, in the CMA-ES results, during the 50% and 75% windows, variable contributions to PC2 and PC3 increase and, in some instances, surpass those of PC1.

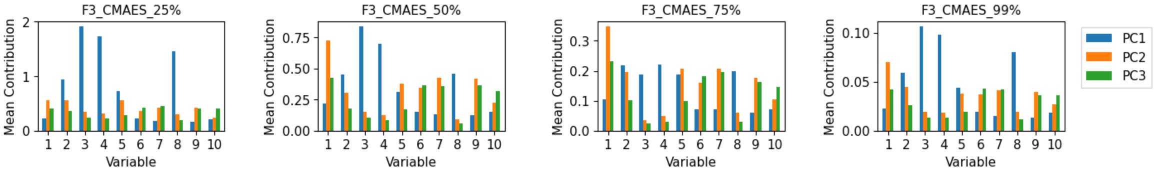


Fig. 6.2 F3 - CMA-ES Mean Variable Contributions

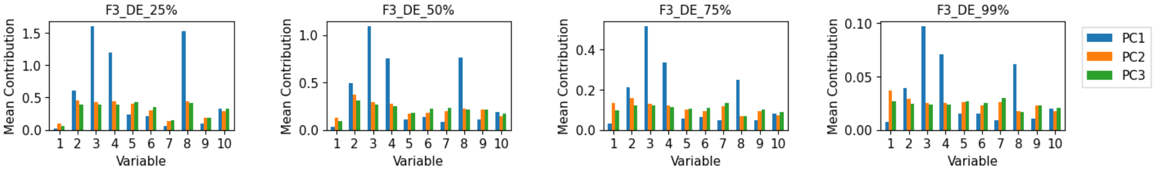


Fig. 6.3 F3 - DE Mean Variable Contributions

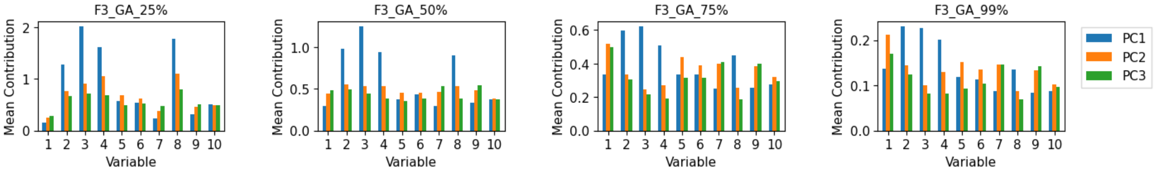


Fig. 6.4 F3 - GA Mean Variable Contributions

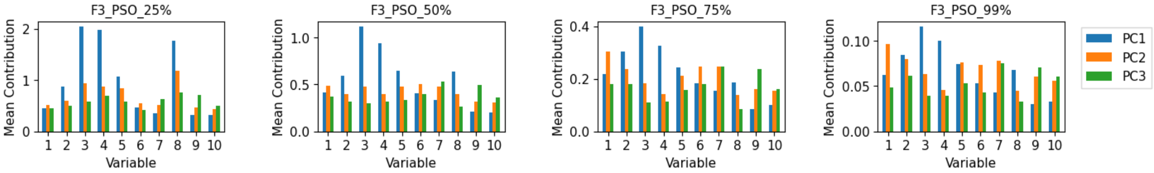


Fig. 6.5 F3 - PSO Mean Variable Contributions

Function F12

Function F12 results show that variables [1, 2, 3, 6] were all found to have contributed highly to the algorithm search paths across PC1 for all four algorithms. Each of these variables' ranks in terms of highest contribution differed between algorithms with variable 1 being the highest for CMA-ES, DE and PSO and variable 6 being the highest for the GA as seen in Figures 6.6, 6.7 and 6.9. The GA's results show that in the 25% window variable 6 followed by variable 2 was the highest, however, the difference in values is very small, with both being approximately 2.5 as seen in Figure 6.8. Interestingly, the CMA-ES results show that in the 75% and 99% windows, variables highly contributing to PC2 were shown to have a higher overall influence than those of the other PCs, especially variable 9. Across the other algorithms, the variables identified as having the highest contribution to position appear to remain higher than the others for the 50% and 75% windows. The PC2 results show some differences between algorithms with variables 2 and 6 contributing more than others in the DE results in all four windows. In the 99% window, all algorithms show a higher proportion of contribution from components other than PC1. This is especially visible in the PSO results.

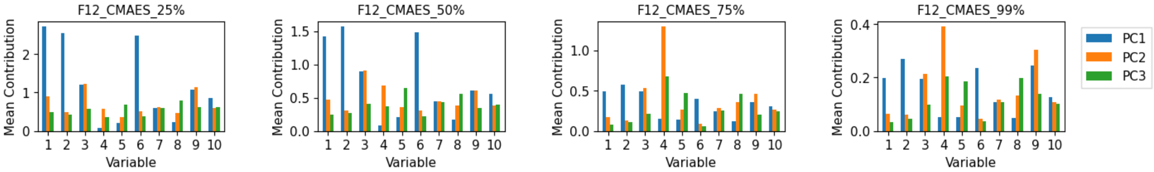


Fig. 6.6 F12 - CMA-ES Mean Variable Contributions

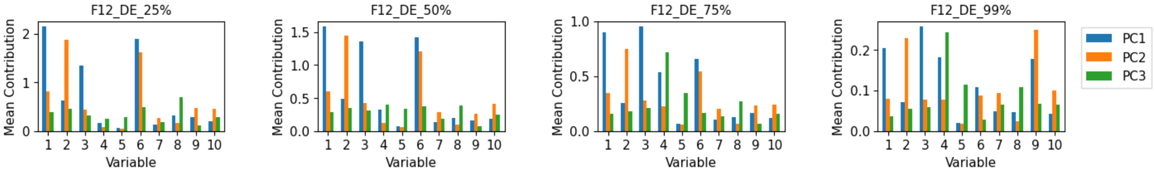


Fig. 6.7 F12 - DE Mean Variable Contributions

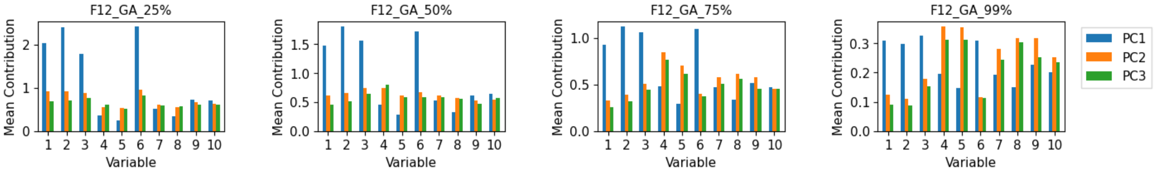


Fig. 6.8 F12 - GA Mean Variable Contributions

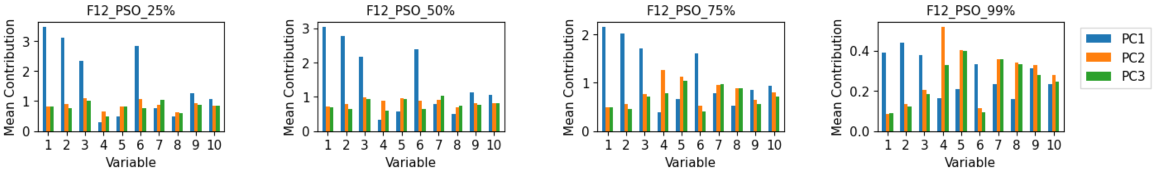


Fig. 6.9 F12 - PSO Mean Variable Contributions

Function F15

The contribution charts seen in Figures 6.10 to 6.13 for function F15 highlight that variables [1, 2, 3, 6] have the highest contribution values across either PC1 or PC2 during the 25% window. The CME-ES, GA and PSO results show these to have the highest contribution across PC1 however the DE results show variable [2] to be considerably more contributory to PC2 than PC1. This remains true in all 4 quartiles for the DE. While these variables are highlighted as being important for all four algorithms, the level of contribution shows variation between them. The CMA-ES, in the 75% quartile, show how variables [4,9] grow in their level of contribution in PC2, becoming the most influential during this stage of the search. The DE results show that variables [1,2,3,4,9] become the most contributory across all 3 PCs by the 99% quartile. The results for DE, GA, and PSO indicate that while initially, a small subset of variables across PC1 has the highest value, by the 99% quartile, the contribution to all PCs becomes more evenly distributed unlike those of the CMA-ES at the same stage.

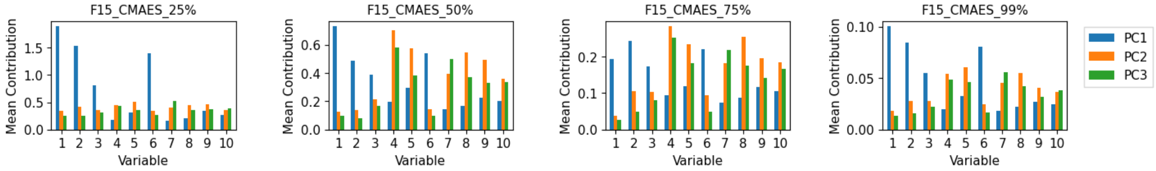


Fig. 6.10 F15 - CMA-ES Mean Variable Contributions

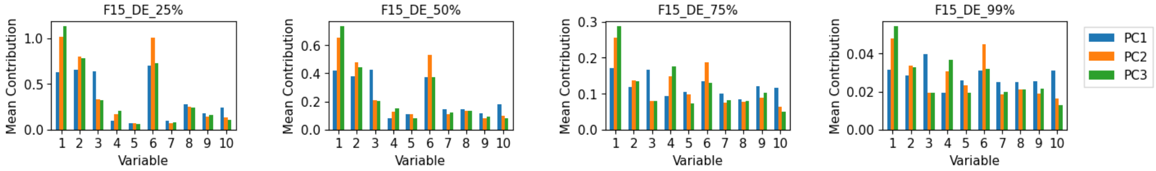


Fig. 6.11 F15 - DE Mean Variable Contributions

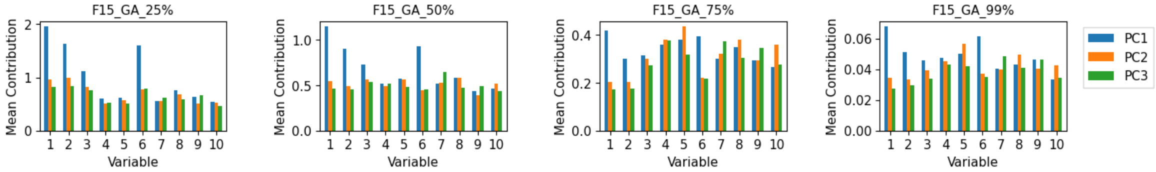


Fig. 6.12 F15 - GA Mean Variable Contributions

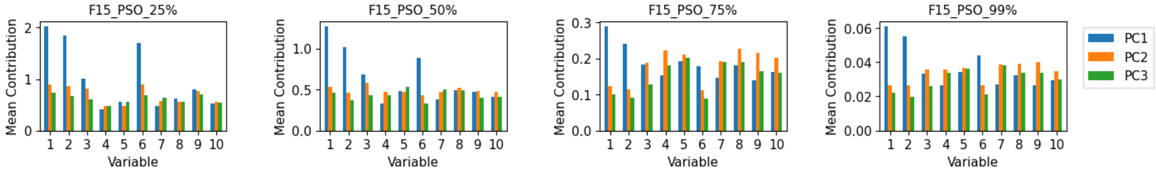


Fig. 6.13 F15 - PSO Mean Variable Contributions

Function F23

Function F23 was selected as an example with a considerably more rugged fitness landscape. The results as shown in Figures 6.14 to 6.17 show the variable contribution calculations in the four fitness windows identified. The results for the CMA-ES, DE and PSO show lower contribution scores across all PCs when compared to the GA, with the DE results showing the lowest values of any in the 25% and 50% quartiles. The problem has the highest mean generation count before 99% of fitness reduction was achieved across all algorithms as seen in Table 6.5. The results show that, in all quartiles for the CMA-ES, GA and PSO, the majority of variables are identified as having a higher contribution in PC1. The more evenly distributed values across all three PCs show that none of the algorithms determined that a specific subset of the variables contributed significantly more than others. Given the longer time to achieve convergence across all algorithms, it may be concluded that no specific subset of variables was driving the search path more than others during each window.

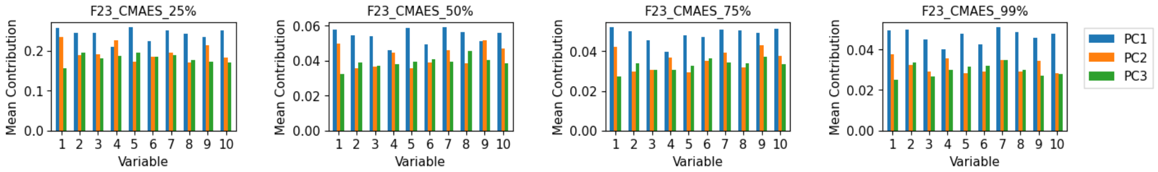


Fig. 6.14 F23 - CMA-ES Mean Variable Contributions

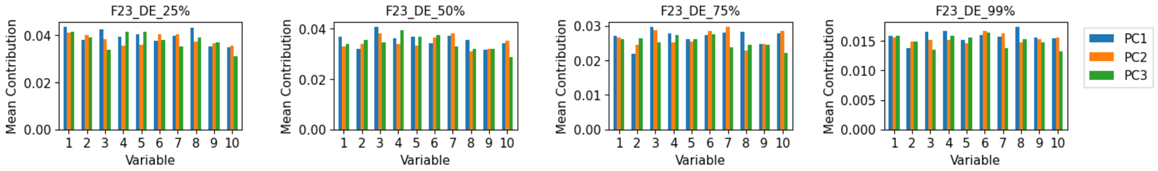


Fig. 6.15 F23 - DE Mean Variable Contributions

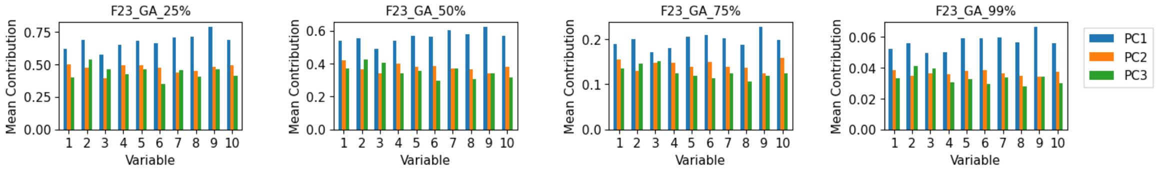


Fig. 6.16 F23 - GA Mean Variable Contributions

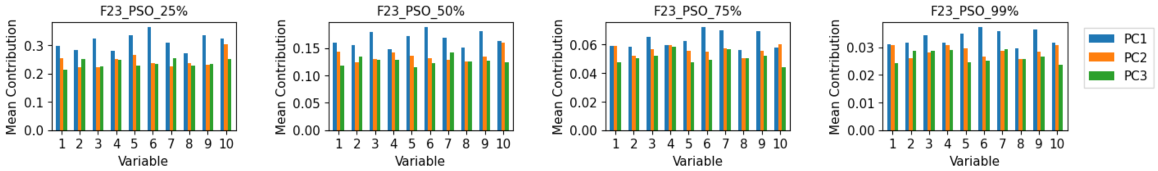


Fig. 6.17 F23 - PSO Mean Variable Contributions

6.4.5 Proportion of Aligned Variables

The Proportion of Aligned Variables (PG_i) is a measurement of the proportion of variables in the input data that are being correctly described by a PC which is a best-fit hyperplane calculated to represent maximal variance in the input data. To calculate this we used Equations

(6.2), (6.3) and (6.4). The input data for this analysis was the extended trajectories so that all trajectories used had the same length. This was done for the purpose of visualising the results so that all plots may have the full range of 300 generations. Alignment is calculated from a solution vector by testing whether the variables are distributed as described by the coefficients of the first three PCs. The more variables that are shown to be in the same direction as the coefficients, the more aligned that solution vector is. This value is calculated for each variable by calculating the proportion of times that it is identified as being aligned in a generation and taking the mean value across all 100 runs.

Function F3

The results of calculating the alignment values for Function F3 are shown in Figure 6.18. Here, we see that the proportion of variables aligned to PC1 for all algorithms begins with a value between 0.65 and 0.85. Shown in red is the natural log of the mean minimum fitness achieved during the optimisation runs for each algorithm. A similar pattern of alignment value changes can be seen in all the results where PC1 begins with the highest value over the first few generations. This quickly falls toward the same values shown in the remaining PCs. The CMA-ES, GA and PSO all show moments in which the variables in PC2 have a higher alignment proportion than PC1 between generations 20 to 30. During this period, more than 50% of fitness improvement has already occurred. The DE results show that while PC1 had an initially higher proportion of variables aligned to it, this value was only 0.68, 0.08 higher than PC2. This value also fell rapidly to match those of PC2 and PC3. The Variable Contribution results show that DE did have a higher contribution score for variables 2 and 6 in PC2 with values close to those highly contributing to PC1 which may explain this behaviour.

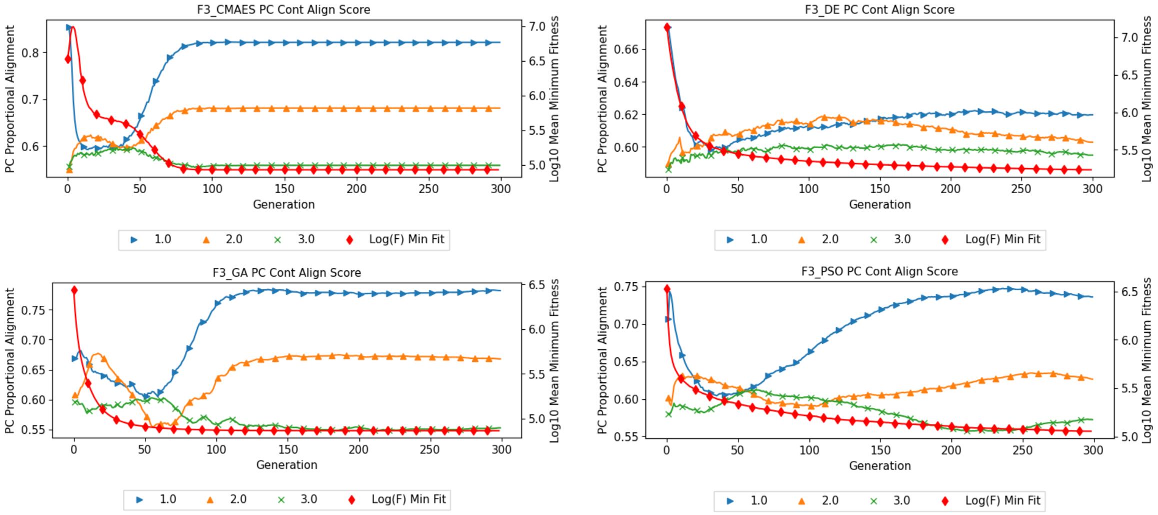


Fig. 6.18 F3 Alignment Proportion Results

Function F12

Figure 6.19 shows the results for Function F12 when calculated across all generations. Here, we see a similar behaviour in both the GA and PSO with an initially high proportion in PC1 that quickly falls before becoming the highest value for the remainder of the trajectory. The DE results show that PC1 had only a slightly higher value than both PC2 and PC3 over the first few generations, all having values between 0.6 and 0.65. When comparing this to the Variable Contribution values, the DE results were the only results to have placed a high contribution to variables across all three PCs in the 25% fitness window in the same manner as was found in the F3 results. The CMA-ES results show an initially high alignment in PC1 with a value of 0.88. This rapidly fell to 0.60 over the course of 3 generations, after which PC2 became the most highly aligned component in terms of variable values. From generations 100 onward PC1 and PC2 alignment proportion values vary however as seen in Table 6.5, the CMA-ES had achieved 50% fitness reduction by generation 4 and 75% by generation 9. This occurs during and shortly after the rapid reduction in PC1 alignment value. By generation 26, 99% fitness reduction had occurred and the erratic behaviour seen after this may be an artifact of the extension of these trajectories.

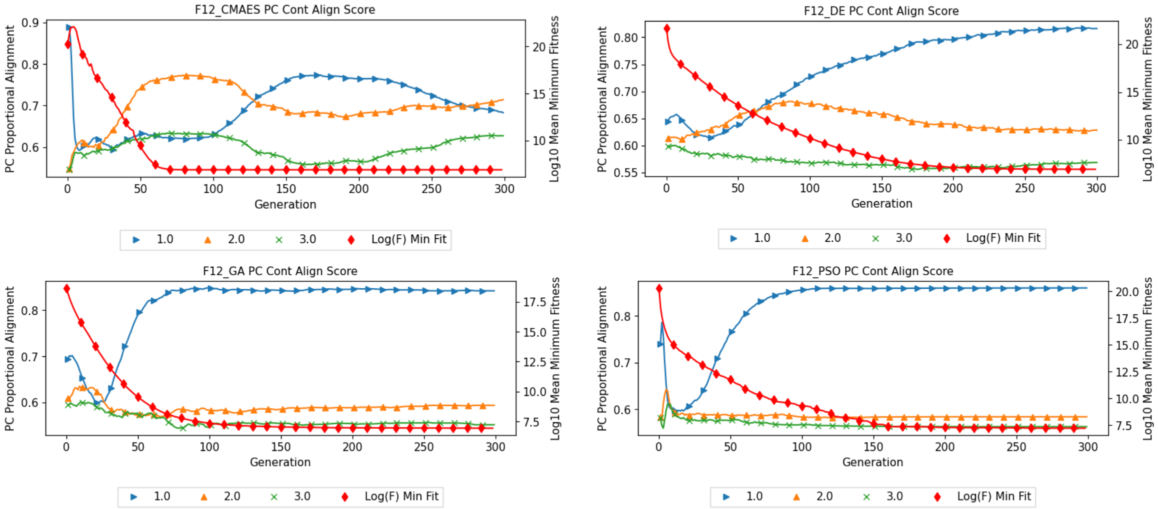


Fig. 6.19 F12 Alignment Proportion Results

Function F15

Function F15 results are shown in Figure 6.20. The GA and PSO results show similar behaviour as PC1 begins with a higher than than the other PCs before dropping as fitness is reduced. PC1 then returns to the highest value for the remainder of the trajectories. The CMA-ES results show a portion of the trajectory, between generations 10 and 50 where PC2 has a higher value than PC1 or PC2. This matches the Variable Contribution results in which we see consistently higher contribution from variables in PC2 and PC3 that exceed those of PC1 during the majority of the run. The CMA-ES 99% window occurs at generation 67, at which point PC1 exceeds all others however this is not fully reflected in the contribution values. The DE results show that overall, PC1 has the highest alignment proportion than the other PCs however as with F3, the range of values is smaller than the other algorithms with PC1 reaching a high of 0.65 and PC3 reaching a low of 0.59. The Variable Contribution results show that the search paths taken by the DE had a higher contribution from a subset of variables that slightly differed from the other three algorithms in PC1. With the smaller proportion value range and higher contribution from PC2 and PC3, this may explain why the DE required a mean of 185 generations to reduce fitness by 99%, greater than the others. Table B.4 indicates that the DE did have a higher contribution in PC1 from the same subset of variables as identified by the other algorithms [1,6,2,3]. The ordering however was different, with the other algorithms ordering these as [1, 2, 6, 3] with [2] being the second most contributing as opposed to [6] for the DE.

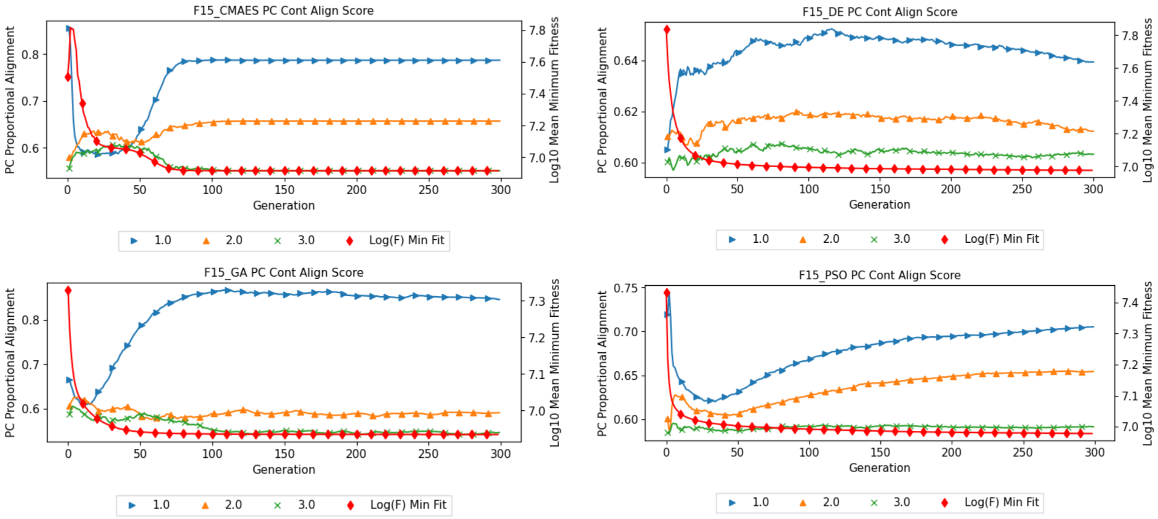


Fig. 6.20 F15 Alignment Proportion Results

Function F23

The results of calculating the alignment values for Function F23 are shown in Figure 6.21. All algorithms show the same behaviour in terms of PC1 starting and remaining the component with the highest level of variable alignment however, the exact value varies between algorithms. The results for the CMA-ES and GA both show a high starting value in PC1. The GA value increases to near 1.0. The CMA-ES value begins at 0.75 before dropping to 0.66 and then increasing to 0.9. For the CMA-ES, DE and PSO the PC2 and PC3 values show little change however the GA values in these PCs decreased overall from 0.8 to 0.6. The Variable Contribution results for all four algorithms show the same pattern where we see a more even spread of contribution values across all variables. Table B.5 shows that while there are some common variables across PCs and algorithms, the same level of agreement on both variable and ranking does not appear as with other problems. One feature of the search that does stand out is the log of Mean Fitness values for the CMA-ES results. Unlike the other algorithms, this value appears to fluctuate slightly but remains high for the first 50 generations before reducing. Table 6.4 shows that the CMA-ES was the best-performing overall with a mean fitness difference of only 0.04 across 100 runs. The fitness windows shown in Table 6.5 show that the CMA-ES had taken slightly less time to achieve a 25% fitness reduction, with a mean generation of 13 however the PSO also achieved this goal within 11 generations. These results may show that the higher mean difference from the optimal solution found in the other three algorithms may have been due to them being unable to exploit the same relationships discovered by the CMA-ES earlier in the search.

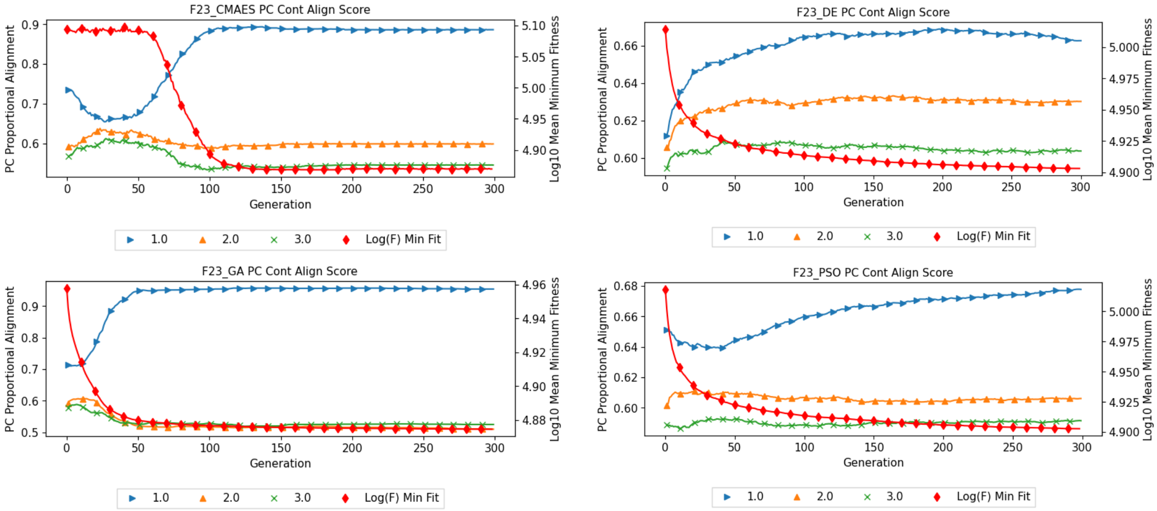


Fig. 6.21 F23 Alignment Proportion Results

6.5 Summary

In this chapter, we presented a set of methods developed to aid in explaining the behaviours of black-box optimisation techniques. The methods introduced were Variable Contribution and Proportional Variable Alignment, designed to mine explanatory features from algorithm search trajectories. These methods are an extension of those introduced in Chapter 3, in which we have adapted the approach to real-value solution representations to detect variable importance at specific stages in an algorithm's search trajectory. This is achieved firstly by identifying the contribution each variable has across a specific component across multiple optimisation runs and secondly, we measure which component, at each stage of the search, is dominant in terms of contribution to fitness change during that stage. Both metrics were designed to mine explanatory features from algorithm search trajectories and as noted in the introduction to this chapter, provide a more interpretable and concise set of explanations regarding algorithm search behaviour. The resulting metrics allow us to create an explanatory model in terms of the contribution these variable subsets have to a given PC, as produced by the PCA decomposition of each search trajectory and taking the mean across multiple optimisation runs.

This linear model represents the relationship between variables as discovered by the algorithm over the course of the optimisation runs and allows us to compare the discoveries of different algorithms. This in turn helps highlight the variables driving the search for higher-quality solutions at different stages in a search trajectory. By generating these explanatory sets of features, in the form of variable contributions, we can help to explain certain aspects of algorithm search behaviour, emphasizing key differences, as evident in the results for CMA-ES and DE. The results also indicate that the subsets identified during the initial 25% fitness window often remain the most influential contributors in subsequent windows, showing that often, variables identified as being highly impactful in the early stages of the search remain so throughout the remainder of the search trajectory. To show this across all 24 BBOB in the dataset, Tables B.2 - B.5 in Appendix B provide a comprehensive overview of the top four contributing variables during the initial 25% fitness window. This not only offers insights into the inherent behaviours of these algorithms but also helps in making these algorithms more accessible and interpretable.

For both User and Developer, these results provide them with key information regarding which variables in their problem predominantly drive the search for high quality solutions. This can be used to highlight variables that require further analysis in terms of why they contribute more than others. As the most highly contributing variables appear to be consistent across a range of search methods, the Developer can use this to ensure that any new EA iteration can exploit the same variables as others to ensure high quality solutions are being found. The variable alignment results further support this by providing the Developer the ability to compare search behaviours over time with other EAs and note how this relates to overall performance.

Future work that could extend the techniques and results shown in this chapter includes our aims to explore the possible broader implications of our findings for the field of black-box optimisation. It may be possible to use the features discovered by our techniques to create algorithms that leverage the findings to direct the search more effectively towards higher-quality solutions. This approach may be done in conjunction with the application of alternative dimension reduction techniques such as those previously discussed. The application of these alternative approaches such as auto-encoders or methods capable of capturing non-linear relationships may provide further insights into algorithm behaviour. Taking a more temporal-focused analysis of the trajectories is also part of our ongoing research. This would involve a temporal analysis approach that considers the sequence of solutions, providing insights into the dynamics of the search process.

Chapter 7

Explainability from Trajectory Mining of Variable Importance Rankings in a Staff Roster Allocation Problem

7.1 Introduction

In previous chapters, we have shown how the search trajectories of EAs can be decomposed and mined for explanatory features relating algorithm behaviour to the fitness function. These chapters have covered the application of our developed techniques to binary string and real-valued vector problem representations, covering a range of bench-marking problems. These results have shown that it is possible to highlight the differing search behaviours of a selection of metaheuristics when solving these problems by mining the search trajectories directly. These techniques have also been shown to be able to highlight specific variable importance to the metaheuristic search process when a fitness gradient exists.

In this chapter, we apply the MCA techniques outlined in Chapter 4 to the search trajectories generated by solving a staff rostering problem. Specifically, we analyse the search trajectories using the Mean Squared Cosine (MSC) which measures the variance captured by each category in each variable. The MCA approach used in this chapter was a necessary step as techniques previously used were reliant on the use of PCA, restricting their applicability to binary and real-value vector representations. The staff rostering problem used in this chapter has been adapted to work in nominal problem encoding. This adaptation has required the inclusion of Multiple Correspondence Analysis to allow for the process to be applied to the vector-based nominal solution representation with categorical variables.

The optimisation problem has been developed to closely match that of a real-world problem supplied by BT while using synthetic data in place of real worker rota allocations, seen in [245]. In that work, simulated annealing was used to generate the solutions necessary for the authors to perform their analysis. As stated in that work "...organisations have different requirements and hence a unique implementation is often necessary for the specific problem scenario.". This also applies to the implementation detailed later in this chapter. A restriction of this problem in relation to our analysis is the need for the generation of successive populations of solutions to create the necessary datasets. To this end, it was decided that a competent GA would be used as a suitable substitute for the simulated annealing in the original work. We chose a GA with a small population size due to the time it takes to compute the fitness of each solution. This smaller population size also brings us closer to the real-world implementation of this exact problem, in which a hill climber is used to find roster allocation solutions due to the problem's complexity and running time requirements. It has been expressed to us during this project that a significant barrier in the real implementation of this system is the time taken to generate solutions due to user expectations and time restrictions, hence why a functionally simple GA with a smaller population size was selected. Another benefit of using the GA is to minimise the gap between our implementation and the BT-specific implementation, making it easier in the future to integrate our approach with their systems.

The optimisation problems explored in previous chapters were selected as they have a known global or set of globally optimal solutions and all have had either known problem structures, such as the pre-defined schemata in the trap-5 problem or landscape features such as the modality of the BBOB problems. The roster allocation problem explored in this chapter was selected as it represents a class of problems commonly found in metaheuristic optimisation - a real-world problem with no well-defined structure. The nature of the problem also somewhat prevents an exhaustive search of the solution space for the true globally optimal solution - with 141 variables and between 12 to 60 possible values in each variable, a conservative estimate using only 12 values results in 12^{141} or 1.4607×10^{152} possible solutions.

This chapter contributes towards the completion of research question (Q4) in which the range of problem representations is expanded. By conveying the generated explanations as a set of variables ranked by their impact on solution fitness and flexibility to be altered, this chapter also contributes towards the completion of objective (O5). This chapter aims to show how these techniques can be used to, firstly, help identify subsets of the rota pool that have a high or low impact on overall solution quality, as detected by a GA solving the problem. Secondly, this chapter presents the results as compared to a well-established method of analysis, the analysis of variance (ANOVA) [246]. By comparing the output of the two approaches we can assess the validity of the results generated via the decomposition of the search trajectories. The results and experimentation in this chapter are an extension of techniques published in [1].

7.2 Twelve Week Minimally Disruptive Roster Allocation

The problem that forms the focus of our case study is a variation of the staff roster allocation problem described in [245, 247]. The Twelve Week Minimally Disruptive Roster Allocation Problem consists of an objective function aimed at minimising the range between the maximum and minimum number of workers scheduled to work on each day of the week, over a twelve-week period. To achieve this, 100 pre-defined roster patterns, each specifying a unique pattern of working hours, were generated based on synthetic data provided by the original problem authors. Each worker receives a randomly chosen subset (size 1-5) of these rosters, varying in duration from 1 to 12 weeks and guaranteeing at least 2 consecutive days off per week.

The problem contains several alterations to the representation and overall complexity from the original. These changes were made to better approximate the real-world version of the problem - this iteration ensures that all optimisation runs used begin at a shared, pre-determined "starting state," representing the initial roster configuration and currently worked week. In this scenario, over time, workers have moved from one roster pattern or working week to another, leading the allocations to drift from the initial solution. A one-off re-optimisation of the allocations needs to be performed to reduce the range value as before. Additional changes were made to mimic minimising disruption to the workforce, a secondary goal alongside balancing resource allocation. To achieve this, workers can switch within their initial roster's weeks, but changing from one pattern to another in their pre-selected sub-pool is limited to a fraction of the workforce (20% in this case). Further alterations include the inclusion of a new soft constraint, designed to detect and minimise occurrences of any worker inadvertently working two consecutive Saturdays due to the one-off re-optimisation of the schedules.

x_1	x_2	x_3	...	x_{n-1}	x_n
12	**33**	15	...	9	45

(a) Sol. Extract

Index	...	32	**33**	34
Rota,Week	...	7,2	**7,3**	7,4

(b) x_2 Roster-Week

Table 7.1 Solution Representation

In this problem, a solution takes the form as shown in Table 7.1a. Here, each worker is represented by variable $x_i = 1 \leq i \leq n$. Its value references a table specific to that variable, containing all possible combinations of their roster sub-pool and starting weeks. Table 7.1b shows an example for variable x_2. In this example instance, $x_2 = 33$ indicates worker x_2 starts the 3-month period using Roster 7 from week 3. They will repeat Roster 7 if it finishes before the period ends.

As in the original problem, we continue to use an “attendance matrix" A_{kd} which, in our 3-month problem, is a 12×7 matrix of the sum of the workers assigned to work on week k, day d. These totals are determined by the variable values in solution X which will in turn determine the total number of workers scheduled for each day d, in all 12 weeks of k. This maps the original problem definition to our trajectory definition. The range between total workers assigned for each day is calculated using matrix A for each column d as shown in (7.1). This contributes to the solution's fitness value in Equation (7.2), subject to a constraint on roster changes shown in (7.3).

$$R_d = \frac{\underset{d}{max}\,(a_{kd}) - \underset{d}{min}\,(a_{kd})}{\underset{d}{max}\,(a_{kd})} \tag{7.1}$$

$$\text{minimise: } \sum_{1 \leq d \leq 7} w_d R_d^2 + \left(\sum_{n \in x} Pe_n \right) S \tag{7.2}$$

$$\text{subject to: } \sum_{i=1}^{n} \text{CV}_i \leq 0.2 \cdot \text{len}(x) \tag{7.3}$$

The cost function shown in Equation (7.2) minimises the overall range in the number of workers assigned to each day, with day-specific weights (w) reducing weekend impact (e.g.,$w = [1,1,1,1,1,10,10]$). This reflects the smaller number of available workers during weekends as stated in [245]. A hard constraint as seen in Equation (7.3) controls roster changes: CV_i represents the total workers assigned new rosters within their sub-pool. This limits such changes to 20% of the workforce. A soft constraint linked to a binary array Pe penalizes two consecutive Saturdays being scheduled due to a roster change. $Pe_n = 1$ if variable n results in this violation, adding a penalty of $S = 0.01$ to the cost function. The aim is to discourage such occurrences while minimising the range for optimal resource allocation. Data files for rosters and allocations can be found at [248].

7.3 Experimental Setup

The trajectories representing runs of a GA ($\mu + \lambda$) with a population form the target of the explanation techniques presented in this chapter with a similar definition as that shown in Chapter 2. Here, μ is the size of the parent population of solutions and λ is the size of the resulting next generation of solutions. These are created through the application of the internal operators of the GA. The Python Multi-Objective Optimisation (PYMOO) library [43] was used to implement the GA with the required parameters. After some initial testing, the values in Table 7.4 were selected to allow for reasonable solution convergence however it is important to note that refining the algorithm's performance was not a consideration in this study. Provided that higher quality solutions were being generated, we are able to continue with our analysis. As this was the case, where possible the default values and operators outlined in the PYMOO documentation were used.

7.3.1 Problem Settings

The instance of the problem used in this chapter has been designed to be minimally disruptive and as such, each optimisation run is initialised with the same starting population of solutions. This population consists of only the starting state solution, the values for which are shown in Table 7.2. This means that any initial changes to solution fitness are achieved by mutation, after which the selection and crossover operators begin to impact the search trajectory of the GA. This decision was made to test whether significantly higher quality solutions could be found that are as least disruptive as possible. By starting at the same point in all optimisation runs, the search is initially limited to solutions in close proximity to the start state of the system.

36	24	2	0	10	10	58	3	11	16
38	72	7	9	0	9	19	46	6	20
4	61	8	10	18	3	31	6	60	64
6	8	10	13	0	0	11	29	10	11
40	35	8	38	27	57	6	51	11	49
18	2	1	10	16	5	14	12	11	29
3	30	26	57	37	25	43	29	4	24
27	10	6	10	36	43	0	34	1	9
6	3	5	9	7	3	46	7	31	2
17	6	0	37	18	25	0	23	9	10
5	39	26	8	0	25	12	37	4	62
9	9	58	5	26	8	54	30	42	34
10	21	1	2	3	10	6	38	29	5
3	60	9	12	21	21	8	29	11	38
25									

Table 7.2 Starting State Solution

Shown in Figure 7.1 is a visual representation of the roster allocations for all 141 staff at the starting state of the problem. In this Figure, blue marks are when each worker is scheduled to work and yellow are days off. The starting state solution used as the base for the optimisation has the fitness and range attributes as shown in Table 7.3. Here, we see that the overall fitness value of this solution is 13.87. The overall range across all 12 weeks has a value of 3.16, showing that the mean range across all weekdays in the 12-week schedule is slightly more than 3 days. The R^2 value of this range, as calculated in Equation (7.8) is 13.08. the solution has a total of 79 out of 141 of the workers having to work 2 consecutive Saturdays, adding 0.79 to the fitness value, shown as the second summand of the same equation. This serves as the baseline for all optimisation runs to improve upon.

Range	Range^2	Saturdays	Penalty	Fitness
3.61	13.08	79	0.79	13.87

Table 7.3 Starting State Solution Details

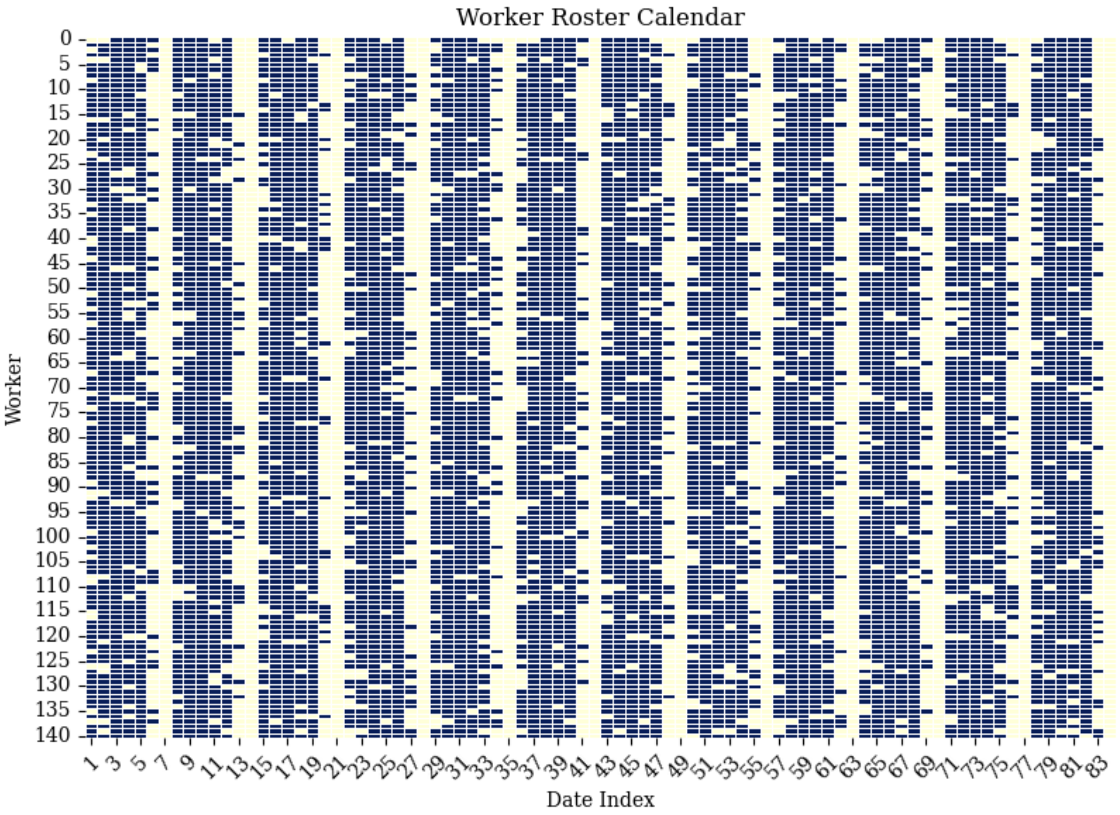


Fig. 7.1 Starting State Rota Allocation

7.3.2 Algorithm Run Settings

Shown in Table 7.4 are the GA settings used when generating the search trajectories that form the basis of the analysis.

n	N	g	Runs	Sel.	Mut.	eta	Cross.
141	20	100	100	Tournament (Size 2)	Polynomial	3.0	SBX

Table 7.4 Algorithm Run Settings

Here, *n* is the size of the solution vector, *N* is the number of solutions in each generation and *g* is the number of generations allowed in each run. The GA was run for a total of 100 optimization runs.

As the mutation used was a polynomial mutation [238] function, details of which can be found in [239], the setting **eta** was set to 3.0. The higher the value is set the more similar and less mutated the child solution will be. The solutions were encoded as discrete variable vectors, in which each value in the vector represented the value given to a specific worker.

These values represented the index of a worker-specific table that contained all possible combinations of Rota and Starting Weeks. This representation required an implementation of the GA that could account for possible disruptions to a solution introduced by the internal operators.

7.4 Analysis Methods

7.4.1 Multiple Correspondence Analysis

As outlined in detail in Chapter 4.1.3, it is possible to link MCA to PCA via the application of an unstandardized PCA to a given indicator matrix. This process allows us to perform a similar level of analysis to the nominal-valued problem instance used in this chapter as would be possible with PCA on real or binary-valued problems as previously shown. Doing so allows the use of the Mean Squared Cosine (MSC) metric as the basis for the calculation of variable contributions, linking this analysis back to the PCA-based approaches in previous chapters. Also shown in that chapter, the strength of any relationship between observed categories within a given nominal variable can be measured using the MSC metric, with higher values implying a stronger relationship. Using this method, we can imply a greater importance of those categories in capturing the structure and variability in the MCA analysis. The equations needed to calculate the MSC are repeated here in Equation (7.4) to (7.6) for reference.

$$PC_{x_{nc}} = \sqrt{\lambda_m} \cdot \text{Factor Loadings}(x_n, p^m) \tag{7.4}$$

$$PC_{x_n} = \sqrt{\sum_{c=1}^{num_c} \lambda_m} \cdot \text{Factor Loadings}(x_n, p^m) \tag{7.5}$$

$$\text{MSC}(x_n) = \frac{1}{num_c} \sum_{c=1}^{num_c} \left(\frac{PC_{x_{nc}}}{PC_{x_n}} \right)^2 \tag{7.6}$$

7.4.2 Analysis of Variance

Outlined in Chapter 4.2.2, here we repeat the calculations necessary to perform the analysis of variance for ease of access, as shown in Equation (7.7) to (7.9). This process is used to generate a complimentary dataset containing the partial-eta (η_p^2) squared values of each variable in the problem.

$$SS_{\text{within}} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)^2 \tag{7.7}$$

$$SS_{\text{between}} = \sum_{i=1}^{k} n_i (\bar{x}_i - \bar{x})^2 \tag{7.8}$$

$$\eta_p^2 = \frac{SS_{\text{between}}}{SS_{\text{between}} + SS_{\text{within}}} \tag{7.9}$$

The partial-eta value, shown in Equation (7.9) is then used to rank the variables, from highest to lowest. ANOVA is performed on each optimisation run independently, resulting in a total of 100 datasets containing the partial-eta and associated $p-values$ values for each variable in each run. In order to create the final rankings for these, certain adjustments need to be made. There are several methods of accommodating this approach that would allow us to use the $p-values$ from across multiple runs to determine the significance of our findings. Our first consideration was the Bonferroni [249] method, however as we are using 100 total runs, this would require us to adjust to *p-value* threshold to a prohibitively small value due to how the Bonferroni correction is calculated. We opted to use the less conservative Benjamini-Hochberg [250] process to instead account for the false discovery rate (FDR) that using 100 runs may introduce. This method, as shown in Algorithm 8 involves the ranking of all calculated $p-values$ before calculating the "Benjamini-Hochberg critical value".

This is then used to determine, for each variable, what the relevant adjusted *p-value* should be to keep the FDR below 0.05 and higher results are rejected and removed from the dataset. Once complete, the resulting dataset can then be used to rank the variables from highest to lowest mean partial-eta value across all 100 runs while ensuring that the associated $p-values$ of those partial-eta measurements are below the necessary, adjusted threshold.

7.4.3 Weighted Ranked Biased Overlap

As we have two sets of variable rankings, each produced from a separate analysis method, it is necessary to incorporate a method to compare and contrast the results of each into the results. As shown in Chapter 4.2.1, the Weighted Ranked Biased Overlap (WRBO) method was selected in this thesis. This approach, repeated here in Algorithm 9 for reference, allows the similarity scoring between two sets of ranked items. The benefits of this approach include the ability to weight the similarity score to subsets of the lists using the internal *p* parameter and that both lists do not need to be the same length or contain the same elements.

Algorithm 8 Benjamini-Hochberg Procedure

```
Require: p-values $P[1..m]$, false discovery rate $Q$
Ensure: List of rejected hypotheses $R$
Arrange the p-values in ascending order: $P[1] \leq P[2] \leq \cdots \leq P[m]$
Initialize an empty list of rejected hypotheses: $R = []$
Initialize the largest index $k$ found so far for which $P[k] \leq \frac{k}{m}Q$ to 0
for $i = m$ to 1 do
    Calculate the Benjamini-Hochberg critical value: $BH = \frac{i}{m}Q$
    if $P[i] \leq BH$ then
        Update $k = i$
        exit for-loop
    end if
end for
if $k > 0$ then
    for $j = 1$ to $k$ do
        Add $j$ to the list of rejected hypotheses: $R = R + [j]$
    end for
end if
return $R$ =0
```

Algorithm 9 Rank Biased Overlap (RBO) Python

```
Require: Two lists $S$ and $T$, weight parameter $wp$ (default: 0.9)
Determine the maximum length $k \leftarrow \max(\text{len}(S), \text{len}(T))$
Calculate the intersection at depth k: $x_k \leftarrow |\text{set}(S) \cap \text{set}(T)|$
Initialize summation term: summ_term $\leftarrow 0$
for $d = 1$ to $k$ do
    Create sets from the lists:
    set1 $\leftarrow$ set($S$[: $d$]) if $d < \text{len}(S)$ else set($S$)
    set2 $\leftarrow$ set($T$[: $d$]) if $d < \text{len}(T)$ else set($T$)
    Calculate intersection at depth d: $x_d \leftarrow |\text{set1} \cap \text{set2}|$
    Compute agreement at depth d: $a_d \leftarrow \frac{x_d}{d}$
    Update: summ_term $\leftarrow$ summ_term $+ wp^d \cdot a_d$
end for
Calculate Rank Biased Overlap (extrapolated):
$rbo\_ext \leftarrow \frac{x_k}{k} \cdot wp^k + \frac{(1-wp)}{wp} \cdot$ summ_term =0
```

7.5 Results

7.5.1 Decomposition Results

The results shown in Figures 7.2 and 7.3 show the mean explained variance, by componant, and adjusted $p-value$ distribution for the MCA and ANOVA analysis respectively.

The ANOVA results show that all included values in the analysis are below the set threshold of 0.05, with nearly all being below 0.003. The Benjamini-Hochberg procedure resulted in approximately 30% of the partial-eta dataset being removed due to the change in $p-value$ thresholds to reject the null hypothesis relating to a variable's effect in solution fitness. This ensured that the variable rankings produced via the ranking of the ANOVA partial eta values remained viable. Similarly, the results of the MCA decomposition were inspected to ensure that the results were representative of the level of variance captured by this method. Seen in Figure 7.2 is the percentage of variance explained by each of the components generated when averaged across all 100 runs. These results show that the first component explains a mean value of around 7.4% of the variance in the dataset. The level of explained variance from component 2 onwards drops off significantly. As the first component explains a relatively small amount of variation, we show the results of using multiple subsets of the components for comparison.

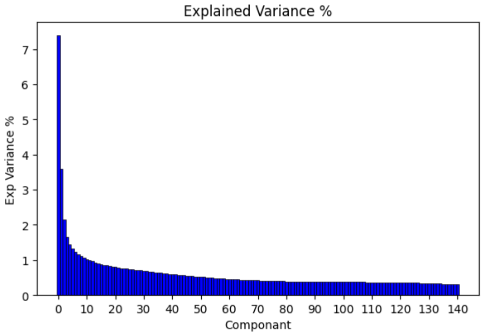


Fig. 7.2 PCA Explained Variance by Component

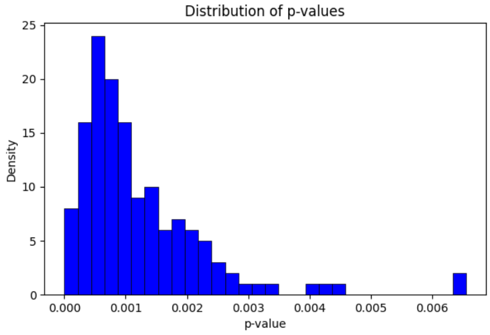


Fig. 7.3 Adjusted Anova p-Value Distribution

7.5.2 Algorithm Performance

The working patterns shown in Figures 7.4 and 7.5 show a comparison between the initial starting state of the problem and a best-found solution in one randomly selected optimisation run. At first glance, there appears to be little difference between the two, with no significant change in weekend cover or alterations to individual working patterns assigned. The starting state solution has a fitness value of approximately 13.9 and the best-found solution has a fitness value of approximately 0.65, showing a significant improvement in overall solution quality.

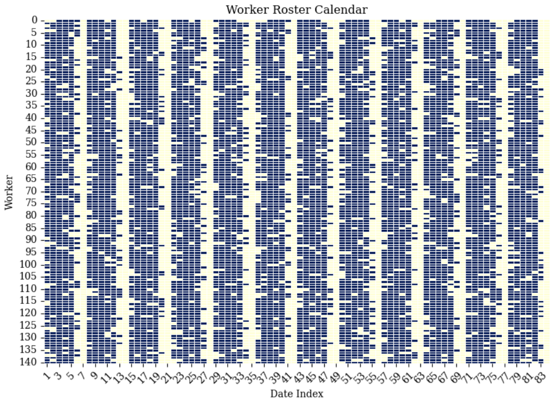


Fig. 7.4 Starting State Rota Allocation

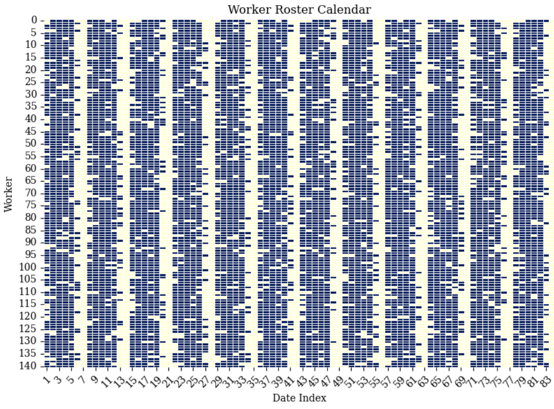


Fig. 7.5 Best Found Rota Allocation.

When comparing the two solutions more closely, Figure 7.6 shows the differences between the two in terms of changes to working days and days off. Areas in red are days that workers are now scheduled to work on when previously they would have been off. In blue are newly assigned days off in comparison to the starting solution. This figure shows that although not visually immediately obvious when inspecting the schedules, there are changes to the overall working days necessary to find a high quality solution. Importantly however, only 37 of the 141 workers have been affected by the optimisation, as seen in the significantly larger number of workers with no red or blue markers indicating a change of working days.

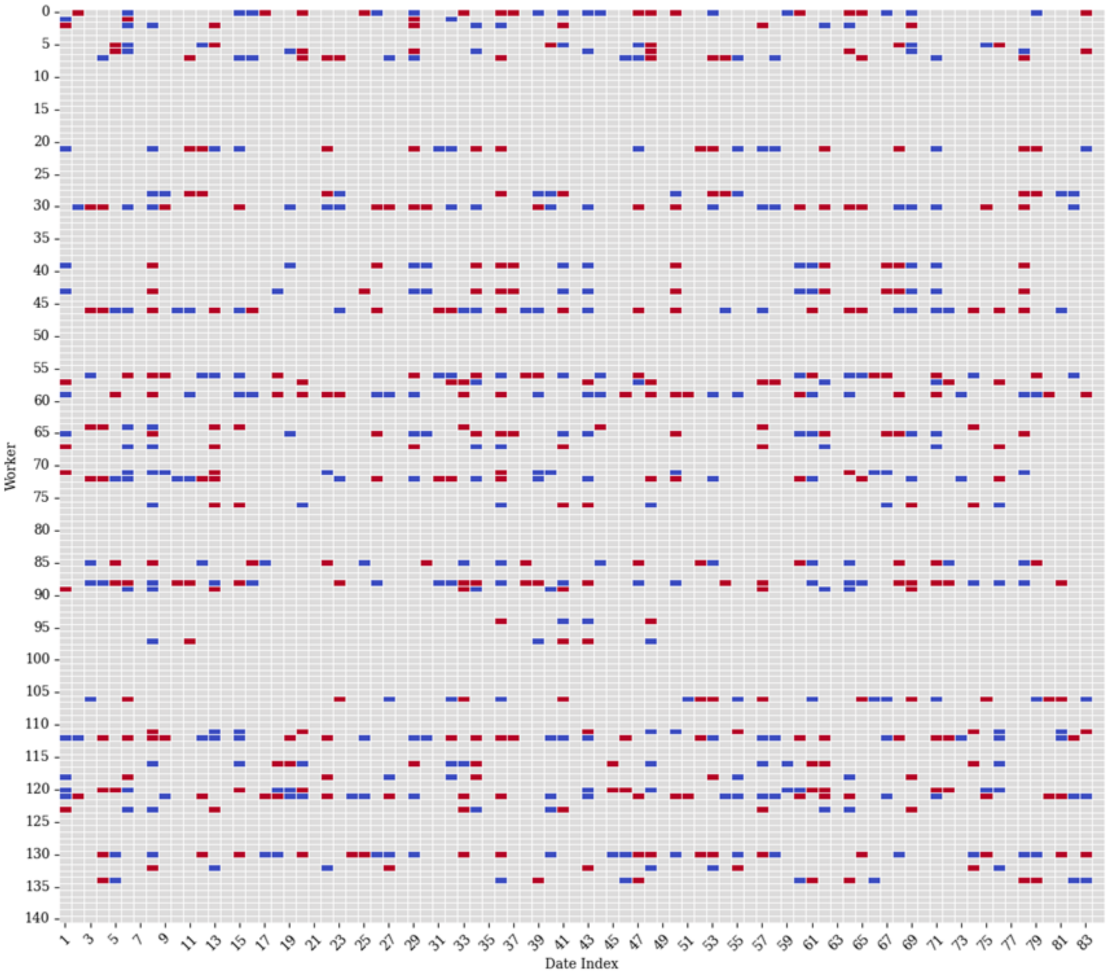


Fig. 7.6 Comparison Between Starting State and High Quality Solution

The results of running the optimization a total of 100 times can be seen in Figures 7.7 and 7.8. These show the mean results for fitness and usage of both the hard and soft constraints in the problem. Figure 7.7 shows the results between solution fitness and the number of variables that were assigned a value resulting in a change of rosters (CV). We see that averaging over 100 runs, the GA was able to find considerably better solutions than the initial starting state of the problem – from a mean fitness of 14 to approximately 0.7. Within the first 100 generations, the value of CV increases from 0 to 6.

Over the course of the remaining 400 generations, this value continues to increase but at a lower rate. This shows that within the first 100 generations, a significant improvement in

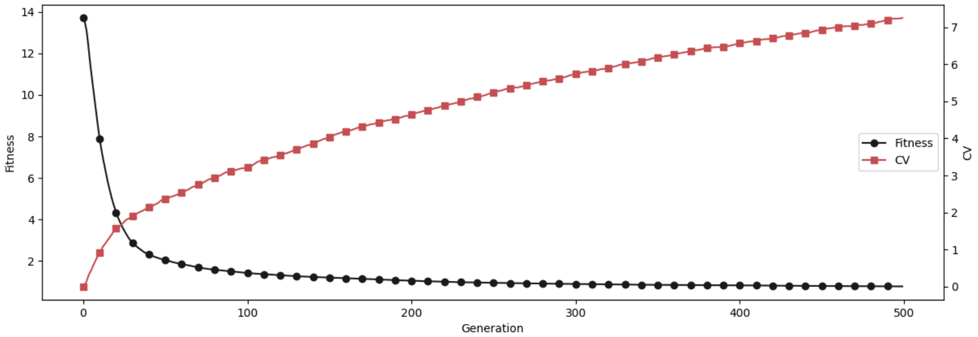


Fig. 7.7 CV vs Fitness

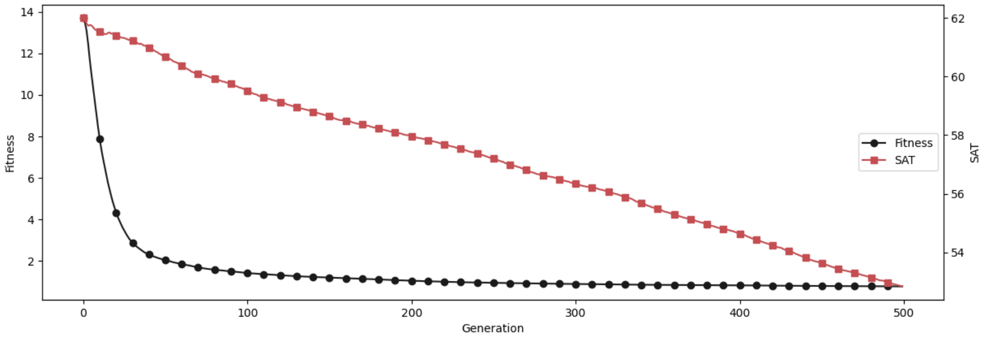


Fig. 7.8 Constraints vs. Mean Fitness

solution quality is achieved with approximately 6 roster reassignments. As the rate of fitness change slows, we see that the GA makes small, incremental improvements to solutions at the expense of adding new roster assignments. Figure 7.8 shows the results of fitness and the soft constraint SAT - the number of consecutive Saturdays assigned. The results show a slow, steady reduction in this value over all 500 generations from an initial value of 62 to approximately 52. Shown in Table 7.5 are the summary statistics calculated from the final population of solutions across all 100 independent optimisation runs.

	Mean	Min	Max	Std
F	0.782	0.654	1.121	0.059
CV	7.262	1	13	2.159
SAT	52.834	44	62	3.524

Table 7.5 Final Generation Results

The changes in daily range values can be seen in Figure 7.9. Here, we show the distribution of range values for each day of the week over all 100 runs. Figure 7.9 shows the range values at 3 different generations - 5, 100 and 500. Between generation 5 and 100 we see a clear reduction in the mean range values across all runs for all days of the week except Saturday, with this day showing a small increase from 0.075 to 0.078. We also see a reduction in the upper limit of ranges seen on Mon, Tue and Wed. Between generations 100 and 500 we see an increase in the mean range on Mon and Sat while the other days show either a reduction or little change. As the fitness value continues to reduce over this period, solutions with a higher range value on some days are being found to achieve a higher quality solution with a more balanced overall range across the week.

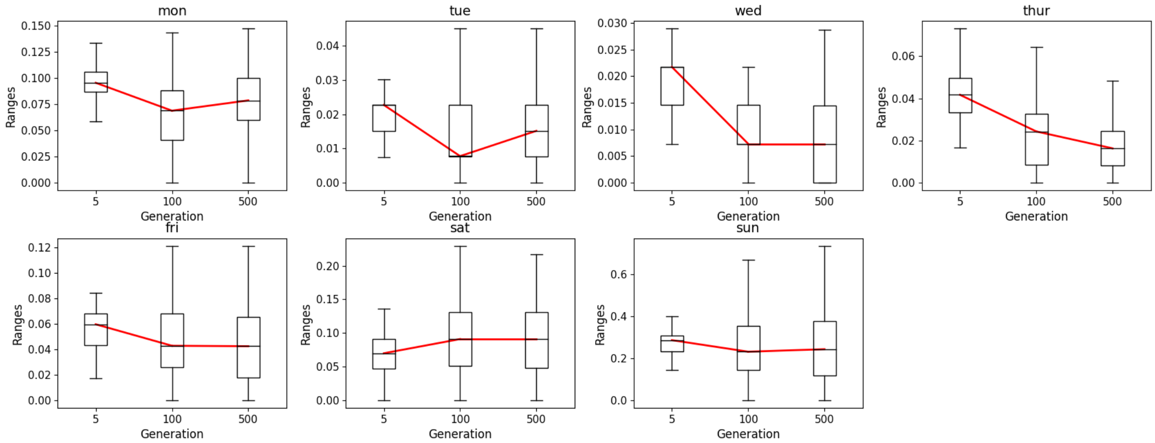


Fig. 7.9 Range Results - Unweighted

7.5.3 MCA and ANOVA Ranking Results

Across all 100 optimisation runs, the MCA's mean MSC value is used to generate the set of variable rankings across a selection of components. Shown in Table 7.6 are the top 10 ranked variables, from 132 to 141, for each of the sets of selected set of components.

The component sets selected all share common members with the previous set. "5PC" contains the first 5 components, "10PC" contains the first 10 components and so forth.

The variable ranking table shows what rank was assigned to the top 10 most influential variables as determined by the application MCA and resulting MSC values across a selection of PCs. **(bold)** values in in each column correspond to a matching variable in the top 10 for that component when compared to the ANOVA rankings which are found in the final column of that table. At the bottom, we show the total number of matches between the two. The results show that again the dataset containing the rankings using only the first PC contains the highest number of matches at 4. Of those matches, variables [121, 65, 60] have an exact rank match to the ANOVA results and variable [1] is only 1 rank away. The number of matches decreases as more components are added to the calculation.

Rank	1PC-MSC	5PC-MSC	10PC-MSC	50PC-MSC	AllPC-MSC	ANOVA-ETA
141	**(VAR_121)**	**(VAR_1)**	VAR_96	**(VAR_1)**	VAR_70	**(VAR_121)**
140	**(VAR_65)**	**(VAR_121)**	**(VAR_121)**	**(VAR_65)**	**(VAR_1)**	**(VAR_65)**
139	**(VAR_1)**	**(VAR_65)**	**(VAR_1)**	VAR_135	VAR_67	VAR_51
138	VAR_96	VAR_96	**(VAR_65)**	VAR_96	VAR_91	**(VAR_1)**
137	VAR_135	VAR_135	VAR_135	VAR_91	VAR_135	VAR_33
136	**(VAR_60)**	VAR_72	VAR_119	VAR_67	**(VAR_65)**	**(VAR_60)**
135	VAR_72	VAR_119	VAR_128	VAR_72	VAR_96	VAR_129
134	VAR_128	VAR_128	VAR_67	VAR_70	VAR_68	VAR_126
133	VAR_48	VAR_48	VAR_91	VAR_48	VAR_76	VAR_110
132	VAR_119	VAR_73	VAR_72	VAR_119	**(VAR_60)**	VAR_66
Matches	4	3	3	2	2	-

Table 7.6 ANOVA to MCA Ranks by Component

(Bold) *Highlights ANOVA to MCA Shared Variables in Top 10*

The results of applying the WRBO similarity scoring process on all 141 variable rankings computed using the MSC values, seen in Table 7.7. Here, we show the similarity score between the rankings generated by MCA when scored against the rankings generated by ANOVA. Highlighted in **bold** are the best found results when comparing the similarity scores across the selection of PCs used.

Dataset	Num. Vars	WP_Val	1PC	5PC	10PC	50PC	AllPC
WRBO	141	0.9	**0.665**	0.451	0.411	0.339	0.246
WRBO_10	10	0.9	**0.606**	0.394	0.367	0.287	0.204
WRBO	141	0.98	**0.694**	0.611	0.586	0.534	0.499
WRBO_10	10	0.98	**0.449**	0.327	0.32	0.219	0.278

Table 7.7 ANOVA to MCA Ranking Agreement (WRBO)

Bold *Shows Best Performer by Dataset*

The results show that for all datasets, the results that most closely match the ANOVA derived ranks are achieved using only 1 PC. This holds true across both sets of 141 variables and the top ranked 10 variables, even when the WRBO WP value is adjusted to place a greater emphasis on the top 10 and top 100 variables. As noted, varying the WP value attributed approximately 85% of the similarity score to the top 10 or top 100 ranks. The highest level of agreement is found when using only component 1 - the highest explained variance component. As more are added, the overall similarity score reduces. This holds true for both $WP = 0.9$ and $WP = 0.98$. The highest level of similarity is found when $WP = 0.98$ and only 1 component is used, giving a score of 0.694, showing the considerable overlap in findings between the two methods.

7.6 Summary

In this chapter we have shown how the search trajectories of a GA solving a staff roster allocation problem can be mined for explanatory features relating algorithm search behaviour to the fitness function of the problem. The search trajectories were decomposed using MCA in order to calculate the mean squared cosine value for each - this determines the relative impact each variable and observed values have in the definition of the resulting components of the MCA subspace. These components define a best-fit hyperplane in the subspace that maximises variance in the observed data. This process allowed us to compute the mean squared cosine of each variable and use this value to create a set of rankings, from 1 to 141, in terms of observed variable importance in the MCA subspace, across 100 independent optimisation runs and using a selection of component subsets. These rankings suggest the relative importance each variable, from the perspective of the search heuristic, had in finding higher quality solutions.

The rankings generated from the mining of the search trajectories were then compared to a similar set of rankings created from the application of ANOVA testing. These ranks were assigned based on the resulting partial-eta values from this process - a measure of the impact on fitness variance that each variable had. By ranking, from highest to lowest partial-eta, we create a complementary dataset. The results of comparing these datasets using the WRBO method for list similarities indicate that with only 1 component, a significant overlap is found between the two sets of results, showing a similarity score of 0.694. One factor in the rankings of the GA, as shown by the ANOVA results, is that the MCA trajectory analysis gives a higher ranking to variables that have an observably higher effect on varying fitness.

Further analysis shows that the rankings that the MCA based method generates indicate that the GA search trajectory is in some way influenced by variable pool size. Seen in Figure 7.10 is the size of the pool each variable has been assigned, compared to the rank assigned by our analysis method. Here, the results show that the larger the pool size, the higher the ranking with a Pearson's correlation coefficient of 0.725, as indicated in the Figure.

This goes some way to explaining the variable importance assigned by the GA, highlighting that larger pool sizes result in higher impact on fitness. This suggests that in the future,

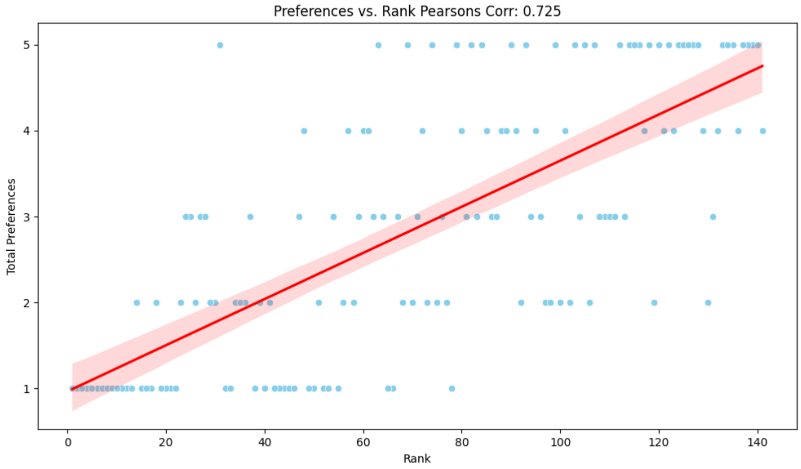


Fig. 7.10 Ranking Vs Roster Pool Size

BT increase the minimum pool size to increase the overall impact that each variable can have on fitness by providing a larger selection of rotas.

Additionally, we can show that the rate of convergence does show a pattern when compared to the variable rankings. While not as strong a relationship, it is clear that some weighting behind variable importance assignment by the GA is due to the rate of convergence. Shown in Figure 7.11 is the distribution of generations at which a variable no longer changes value across all 100 runs. Here we see clearly that there is a distinct pattern of variables taking longer than others in the search before convergence occurs.

When this behaviour is compared to the MCA ranking results, we see that there is a detectable relationship between these two. Here, we see that there is a Pearson's correlation coefficient of 0.68, indicating a strong positive correlation between the mean convergence generation of each variable and its associated ranking.

The combination of the MCA-derived techniques and the use of ANOVA in this chapter provides an enhanced set of results to present to both User and Developer. The use of ANOVA acts as statistical validation for the MSC based rankings as can be seen in the results, in which a high level of agreement between the two methods is found in the high and low impact variables that are consistently identified by both measures. The use of both methods in combination allows the Developer to validate the impact of identified variables, while the User gains confidence in the results through the statistical evidence supporting the rankings.

In terms of explanatory feature detection, the techniques shown in this chapter provide the Developer with insights into how the roster patterns impact the fitness function. This, in combination with the observable influence of the pool size and convergence rates offer the Developer key information regarding algorithm performance. By explaining what the key

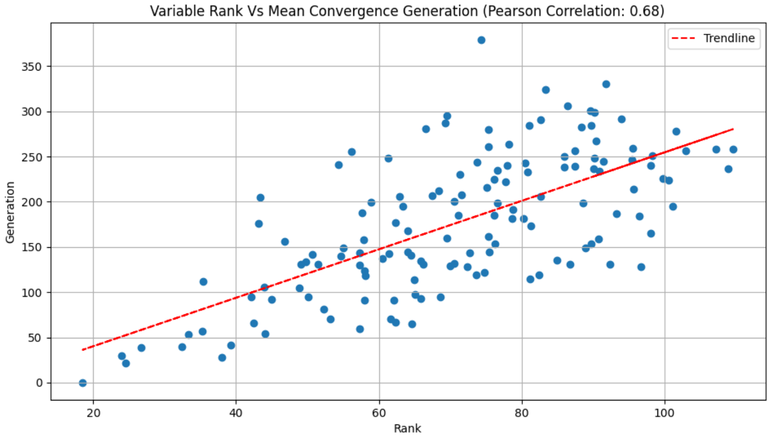


Fig. 7.11 Ranking Vs Mean Variable Convergence Generation

drivers to solution quality are and how they are impacted by pool size and convergence rates, the Developer can fine-tune their algorithms on this problem instance to take full advantage of the findings. Developers can optimise the algorithm's efficiency and effectiveness, ensuring that it targets the most influential variables.

The User benefits from the explanations generated in this chapter by gaining a clearer understanding of which variables influence solution quality – in this case sets of working hour patterns. The ranking system offers an interpretable set of results, making it easier for the User to make informed decisions about variable adjustments. By identifying high-impact variables, Users can focus on maintaining or optimising these, while understanding which low-impact variables can be modified with minimal disruption. This knowledge is particularly valuable for operational decision-making, allowing for more effective resource allocation and improved overall system performance.

In summary, we show that we can mine the search trajectories of a population based search heuristic, in this case a GA, to generate a set of explanatory features that help in understanding the motivation behind the search patterns of the GA to the fitness function. We can show that there are at least 3 main drivers that directed the search toward higher quality solutions in this problem instance - observed impact on fitness variation, the size of the rota pools and the convergence rate of each variable. All three of these aspects show a strong relationship to the rankings generated by MCA, implying that these three are exploited by the GA during the search. Our experiments also identified key variable subsets, corresponding to individual workers, using both the MCA approach and ANOVA testing on the same search trajectories. The overlap between the two methods in the top and bottom rankings provides a subset of variables that have either a high or low influence on the search path and fitness impact. This gives end-users a way to identify key individuals and those who, due to their lower impact, could be moved to another observed rota schedule with minimal disruption. This tool is valuable to end-users, helping to explain key drivers towards high-quality solutions and the capacity for minimal impact change.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

8.1.1 Research Questions

In the writing of this thesis, we aimed to address a set of questions surrounding the development and application of XAI techniques to the search trajectories created by a range of evolutionary algorithms. Here, we discuss each question and to what ends this thesis has addressed them.

(Q1) **In what ways can and do EAs benefit from XAI?**

In chapter 2.2 we review the stochastic internal mechanisms of a selection of evolutionary algorithm approaches. This question also motivates chapter 2.3 in which a selection of XAI techniques are discussed as well as the changing taxonomy in this area. The EAs explored included Vector-based algorithms (PSO, CMA-ES, DE), history-based algorithms (PSO, DE) and Fitness model-based algorithms and strategies (EDA, CMA-ES). Also in this chapter, we discuss the common usages of these algorithms in industry as well as some of the potential impacts these technologies are capable of. While each of these approaches have become the basis for many adaptations in recent years, we have selected this subset of rudimentary evolutionary algorithms for our analysis.

This subset of EAs represents a variety of metaheuristic methodologies while remaining approachable and relevant to both research and industry. The results of our analysis have shown that across binary-string, real-value and nominal vector solution representations, there exist clear algorithm-specific search behaviours that are identifiable by both existing and novel metrics. This highlights that the subset of algorithms selected represents a range of search behaviours on the same optimisation problem and serves as an excellent collection to target for the direct application of XAI methodologies.

(Q2) **How can we formalize a method of solution-population structuring such that novel XAI features can be extracted from the algorithm search path?**

The representation of a population of solutions can, as shown in Chapter 3.3.1, be explicitly mapped to a collection of vectors in the search space of an optimisation problem. We have shown that, provided that the resulting search space of an optimisation problem follows the axioms of a vector space, a collection or generation of solutions in $\mathbb{R}^n$ can be represented as $\mathbb{R}^{gNn}$ where g is the generation that the solution exists in, N is the size of the population and the problem has a dimension of n. This mapping allows us to formalise the structure of a search trajectory in this thesis as seen in Equation (3.6).

$$\begin{aligned} T &= [X_1, \dots, X_g]^\intercal \\ X &= \{x^1, \dots, x^N\}^\intercal \\ x &= [x_1, \dots, x_n] \in \mathbb{R}^n \end{aligned}$$

This formalisation allows us to define a search trajectory - the collection of solutions X visited during an optimisation run - as a list of solutions ordered by g and gathered into a search trajectory T. We also show in Chapter 3.3.5 that this formalisation can be used to map the structure of a search trajectory to the creation of the set of variance-based best-fit hyperplanes - principal components - created via PCA.

(Q3) **Can the search trajectories of EA, across a known fitness gradient, be mined for novel explanatory features that relate some level of knowledge regarding algorithm search behaviour to an end-user?**

The results in this thesis demonstrate that PCA effectively retains geometrically significant features of the search trajectories, even after the data has undergone decomposition and dimensionality reduction. Due to its non-destructive and non-additive nature, any discernible structure in the projected dataset is a reflection of the inherent structure in the original data. This core ability of PCA to help isolate and highlight this latent structure allows us to perform data mining on evolutionary algorithm search trajectories. This has been done to detect any structures or relationships with explanatory power relating to an algorithm's search behaviour and the nature of the problem being solved and relate this back to any end-user. As shown in this thesis, this process can produce a subspace with sufficient structure to allow the creation of novel, angular-based metrics capable of capturing a similar level of information to the more traditional KLd measure of population diversity. In addition to this, it has been shown that the coefficients of the resulting components reflect some of the low-level variable interactions that define the fitness function of a set of binary representation optimisation problems. The results in [6, 2] show our research in this area.

(Q4) **Can a method of explanatory feature mining from the search trajectories be formalised and be made applicable to a range of optimisation problem representations?**

In Chapters 6 and 7 we show how the initial methods of explanatory feature mining from search trajectories can be extended from binary to both Real-value and nominal problem representations. To achieve this, we have formalised a method of variable importance assignment by applying a ranking system to the results of the data mining of the search trajectories. In those chapters, we have focused on two specific aspects of PCA - variable contributions and, in the case of nominal representations with categories rather than discrete values, the mean squared cosine metric.

When applying PCA to the search trajectories, we show that we can derive a variable's importance and assign a rank to that importance by calculating the variable contribution as shown in Equation (4.5) and shown here.

$$C_i' = \sum_{k=1}^{N} \left(\frac{\left(\sum_{j=1}^{n} p_j^i x_j^k\right)}{N} \right) p^i.$$

Here, C_i' belonging to $\mathbb{Z}$, is a vector of mean variable contributions across any given component p^i of a generation of solutions with size N and dimensionality n. The formalisation of this approach is introduced in [7]

In Chapter 7 we extend this to the nominal-valued case study problem of the Minimally Disruptive Rota Allocation problem. To accommodate this new problem representation, the use of MCA was required. This change required the use of the MSC metric as calculated in Equation (4.11), (4.12), and (4.13) and repeated here for reference.

$$PC_{x_{nc}} = \sqrt{\lambda_m} \cdot \text{Factor Loadings}(x_n, p^m)$$

$$PC_{x_n} = \sqrt{\sum_{c=1}^{num_c} \lambda_m \cdot \text{Factor Loadings}(x_n, p^m)}$$

$$\mathrm{MSC}(x_n) = \frac{1}{num_c} \sum_{c=1}^{num_c} \left(\frac{PC_{x_{nc}}}{PC_{x_n}} \right)^2$$

By using a ranking system on the resulting variable importances generated by these two methods, we can generate a more interpretable set of explanatory features - relative variable importance to fitness that can be compared between algorithms and additional analysis techniques including the partial-eta based ANOVA method. Further to this, we show how these techniques can be applied to subsets of trajectory generations to gain insights into variable importances at differing stages of the search as shown in [1].

8.1.2 Research Objectives

In the process of answering the research questions in this thesis, it was necessary to outline a series of objectives to aid in this research. Here these objectives are paraphrased and, for each, the progress and outcomes are discussed.

(O1) **To identify a subset of widely used evolutionary algorithms with a representative variety of search mechanisms. This objective aims to address the research question (Q1).**

To answer the first research question, we introduce in Chapter 2.2.5 a discussion of the commonly used evolutionary algorithms in industry which served as the selection process for those used in this thesis. We settle for the following selection of population-based evolutionary algorithms:

- Population-Based Incremental Learning Algorithm
- Genetic Algorithm
- Particle Swarm Optimisation Algorithm
- Differential Evolution
- Covariance-Matrix Adaptation Evolution Strategy

This selection addresses the requirements of (Q1) in that it represents a variety of search mechanisms and as shown in the literature review in 2.2, are commonly found in industry and in academia.

(O2) **To develop a definition that formalised the structure of EA search that can be used to structure future experimentation and analysis and is a move towards**

answering research question (Q2).

This thesis approaches the issue of extracting explanatory features from search trajectories by applying a set of both traditional and novel data mining techniques. To achieve this, a standard structural definition of what a search trajectory is and how it can be incorporated formally into the definitions of analysis techniques was needed.

As noted in how we have addressed the research question (Q2) directly, we developed a search trajectory definition that allows for any problem definition. By using this definition we can map a search trajectory to the resulting projected set of solutions in PCA-derived subspaces directly as well as show how the transformation can be applied using the same language and terms. As the definition is structured by generation it has the added benefit that we can link the novel methods developed in this thesis to the search trajectory definition in terms of collections of generations used in the analysis with no confusion as to their source.

(O3) **Perform exploratory analysis and data mining on the search trajectories created by EAs. This objective is designed to aid in answering (Q3)**

The work in this thesis addresses this objective as shown in Chapter 5 where the initial, exploratory research is outlined. This work was performed on binary search spaces to discover whether PCA was a suitable tool for the analysis of search trajectories as well as whether any explanatory features were detectable in the resulting projection of the search trajectories. This thesis presents the results of this analysis showing that sufficient structure remains post-projection and dimension reduction that, even when represented as a single vector, the cosine similarity measure is capable of capturing a similar level of population diversity information as KLd when analysing the movement of each generation of solutions. The work done in Chapter 5 served to both complete objective (O3) and provide a basis upon which the later work in this thesis builds.

(O4) **Identify a subset of possible end-users that would be most suitable to target the generation of explanations towards. This objective is also designed to aid in answering (Q3)**

This objective was set as, during the early stages of research for this thesis, it became apparent that in the field of XAI end-users and stakeholders of AI systems have varying requirements and expectations. Outlined in Chapter 2.3.1 we show the results of our research regarding what we consider to be the main stakeholder categories. Of these, we highlight two specific stakeholder types which can be considered our intended end-users. Firstly, we outline the User - they typically represent a user with more specific expectations regarding solution quality rather than overall algorithm behaviour. We consider this user to require explanations regarding why a solution has a specific quality and what the main drivers of solution quality are. The second end-user identified as the target of our work was the Developer who typically will represent an algorithm developer in either academia or industry. These end-user requirements tend towards information and explanations regarding algorithm performance and overall search behaviour in comparison to others.

(O5) **Develop a set of methods capable of leveraging any features identified and transform the knowledge gained into a level of explanation suitable for the targeted end-users. This objective is designed to aid in completing (Q3) and (Q4)**

This thesis develops a set of analysis techniques designed to extract explanatory features from the evolutionary algorithm search trajectories and represent them in a suitable format for our intended end-users. Firstly, we show how we can help explain to Developers the distinguishing features of their algorithm's search behaviour as shown in Chapter 6. This set of explanatory features takes the form of changing variable importances and differing levels of variable alignment. From this, we can show to the Developer what variables in the search are driving their algorithm towards better quality solutions and how this changes over time in comparison to other metaheuristics. To aid the User, we show in [7, 1] how similar metrics can be used to create a more interpretable, solution quality-specific set of variables with either high or low impact. We also compare these in the chapter to the AVOVA results to ensure alignment with other analysis techniques. These smaller, more interpretable explanatory features address the Users specific needs shown in Table 2.4 of simple, clear explanations regarding AI output.

8.2 Summary of Thesis Findings

This section aims to summarise the overall contribution this thesis has produced over the course of the research project.

In binary spaces, it has been demonstrated that mining trajectories for explanatory features, derived from the eigenvector coefficients after decomposition, can reveal low-level variable interactions and problem structures significantly influencing data variance throughout optimisation. This variance influence mirrors their effect on solution fitness and has the potential to reveal crucial problem structures for achieving high-quality solutions. Moreover, by representing the solution population as a singular vector in a reduced dimension subspace, it is possible to mine the trajectories in such a way that a metric akin to entropic divergence can be developed. This approach allows the differentiation of algorithms and their search behaviours on identical problems, utilizing angular base metrics created for use in the subspace.

Extending this approach to real-value solution vectors and benchmark problems, the search trajectories can generate explanatory features when mined using an adaptation of these techniques. Firstly, we show that by using the concept of variable contribution to the variance-based subspace's axis definition, it is possible to identify variables that impact the search for higher quality solutions more than others across a subset of these axes. High-contributing variables are more precisely represented by that axis, allowing attribution of data variance changes to these variables. Similar to the binary results, variables or variable groups with substantial influence on variance are highlighted by the decomposition process in such a way that they have a high impact on the search direction which is towards higher-quality solutions. A further adaptation of this method enables the examination of the proportion of highly contributory variables in alignment with the algorithm's direction over the search space at any given moment.

This reveals which variables contribute more across certain hyperplanes, particularly the first component, at specific generations or points in the trajectory. In terms of explanations, these two approaches allow us to highlight variables that are consistently important across multiple runs and different search methods. We can also use these properties to show how, at differing stages of the search, the impact of these variables changes over time.

Finally, the variable contribution methodology is adapted for nominal space, allowing its application to a near-real-world problem. Using the MSC value to calculate variable contributions, this method effectively highlights variables crucial to variance and, consequently, the fitness of a solution consistently across multiple runs. These findings are corroborated by overlapping results with ANOVA, following the Benjamini-Hochberg adjustment. The explanatory features extracted from this approach are subsets of variables with either high or lower-than-average impact on overall solution fitness. The results suggest that the EA implicitly considers variables (roster pattern collections) that are larger than others, with a size of 4 or more being preferable. Additionally, the results also show a tendency for variables to be more highly ranked in terms of importance when they have a later overall convergence generation, suggesting that while some variables contribute to rapid quality gain in the initial generations, the EA puts a higher rank on those with a greater potential for variability. This coincides with the higher rankings given to variables with larger roster pools, providing a larger and more varied pool of options to select from.

8.3 Reflections and Limitations

This section outlines and discusses the main limitations or external factors that impacted the project's scope.

8.3.1 Real-World Data

In Chapter 7 we present the Minimally Disruptive Rota Allocation problem, based on the previous version of this problem produced by BT. In this work, we state that we use synthetic data created using a tool provided by some of the original authors of the problem. This project intended to extend this problem to use real-world data, in this case, real working patterns and allocations to workers in a specific geographical location. Two main factors prevented this extension from being undertaken.

Firstly, the complexity of the actual implementation of this problem in BT was considerably higher than initially considered. As mentioned in Chapter 7, computation time was a limiting factor at BT due to this complexity. Because of this, a more simplistic version was developed and used. The specific BT implementation was also not designed with trajectory output in mind. As such, the current system does not output any of the required trajectory data and no budget for a software engineer's time to address this was available.

The second factor was the availability of real-world data. Due to the timeline of the project, it was decided that the synthetic data generation tool provided would be suitable as gaining access to the real data would require significant time and would require processing and anonymising, adding additional overhead to the project. It is still intended that a more complex version of this problem using multiple geographic locations and more realistic rota patterns be developed and this is being considered for future work.

8.3.2 Single Decomposition Method Reliance

Early research suggested that PCA was a suitable method of decomposition for the project. It was intended that additional methods would be introduced for comparative analysis however this was not achieved. After researching additional possibilities, the method that was considered to be used - t-SNE - was deemed inappropriate due to its stochastic nature and tendency to alter geometrically sensitive structures such as solution distances. As outlined in Chapter 3.3.3 there exist many different methods available. Currently, this thesis only explores the applicability of PCA to the data mining of search trajectories however this limitation is addressed in the Future Work section as it is considered a significant and promising area of future research.

8.3.3 End-User Study

A significant limitation to the scope of this thesis was access to real-world end-users during the development of the methods outlined. Due to uncontrollable circumstances, a significant proportion of the research project was undertaken during a time in which few end users were available. To address this, we realigned the output of our methods towards the two stakeholders identified in Chapter 2.3.1, the Developer and the User, and aimed at generating explanations in as simple a form as possible. This decision led to the development of the ranking system and the use of the WRBO metric for ranked list comparisons. By doing so we were able to design the output of our analysis to closely match our interpretation of these end-users requirements in terms of explanation forms. It is hoped that in the future, access to a selection of end-users can be made available to allow an end-user study to take place regarding the BT roster allocation solutions and explanations generated in Chapter 7 and future collaborations.

8.4 Future Work

8.4.1 Additional Decomposition Methods

There are many ways to approach the decomposition of algorithm search trajectories as outlined in the literature review of this thesis. While there are many methods available, in the area of EC-XAI and XAI in general, many of these techniques are often only used within the "Data Explainability" aspect of XAI. There is a clear need to extend the work outlined in this thesis and in the XAI area in general with additional insights into the capabilities of these decomposition methods in terms of identifying key patterns within EA search trajectories.

A promising option that would suit future work in this area is the direct application of Uniform Manifold Approximation and Projection (UMAP) to EA search trajectories. While this method does share the use of stochastic gradient descent to minimise the cross-entropy difference between the graph representations of the projected and original datasets, UMAP excels at preserving both local and global structure as opposed to the locally-focused t-SNE. This preservation of local and global structure would provide a highly interesting comparison to PCA which is predominately globally focused as well as UMAP's ability to capture complex linear and non-linear variable relationships, unlike PCA which presumes that all relationships are linear.

A significant aspect of this future work will be the development of novel methods of variable importance mining from the resulting UMAP projections. Two possibilities to consider are the use of sensitivity to fitness metrics such as those generated by sensitivity analysis and ANOVA and experimentation to discover if the UMAP's central cost function can be mined for information. For the former suggestion, it may be possible to infer the importance of a variable from the detectable effect on the overall UMAP projection that varying the input data generates. As UMAP uses stochastic gradient descent, perturbing the values of each variable and measuring the magnitude of the change in overall mappings may indicate a variable's overall impact as well as any disruption to structures detected before the alteration.

For the latter suggestion, it may be possible to record differences in the cross-entropy values between the original data and the projection using data from different stages of a search trajectory. There exists cross-entropy and perturbation-based approaches to optimisation and solution approximation however it may be possible to generate explanatory features from a form of "entropy-trajectory" mined directly from the mappings generated by techniques such as UMAP.

$$UC = \sum_{i,j} w_{ij} \log \frac{w_{ij}}{e^{-d(y_i,y_j)}} + (1 - w_{ij}) \log \frac{1 - w_{ij}}{1 - e^{-d(y_i,y_j)}}$$

Shown here is the UMAP cost function (8.4.1) designed to minimise the cross-entropy differences between the two graph representations of the data. As this is a minimisation problem, the entropy-trajectory may consist of either the overall cost value generated by the iterative gradient decent solver (C) or may contain additional information such as the terms w_{ij} which are the weights applied to edges in the graph representation between points i and j, tracking how these weights may change over time.

8.4.2 Surrogate Modelling Trajectory Analysis

Surrogate models [251, 252] are usually used to speed up an EA by replicating a costly fitness function and replacing calls to it with calls to a much cheaper model [253]. Their work noted that the surrogate represents an explicit model of the population, and proposed mining this model to capture the sensitivity of the objective function to the problem variables. The idea is that the model is biased by the population of the EA, and so represents another view of the algorithm's understanding of the problem. Our early work in this area can be seen in [2] in which this line of inquiry was started. This area of research is worth investigating to further expand on the initial results published. This process involved the creation of a surrogate model trained on the final set of solutions generated by a GA, which would then be mined for variable importance. This process, shown in the following algorithm for binary problems could be adapted to allow for a more rigorous exploration of this concept.

Algorithm 10 Method for probing variables in a solution with respect to the surrogate fitness function

```
**In:** $x = (x_0 \dots x_n), x_i = \{0, 1\}$, near-optimal *seed* solution found by GA
**In:** $S(X) \rightarrow f$, surrogate fitness function to estimate fitness $f$ of a solution $X$
**Out:** $C = (c_0 \dots c_n), c_i \in \mathbb{R}$, absolute change to surrogate fitness for each variable in $x$
$C \leftarrow \emptyset$;
$f_{org} \leftarrow S(x)$ {surrogate fitness of solution}
**for** each variable $x_i$ **do**
    $i = 0$ to $n-1$ $x_i \leftarrow (x_i + 1)$ mod 2 {flip variable $x_i$} $\hat{f}_i \leftarrow S(x_i)$ {surrogate fitness of mutated solution} **if** $x_i = 1$ **then** $c_i = -1 * abs(f_{org} - \hat{f}_i)$ {Change in surrogate fitness, optimum should have a 1}
  **else**
    $c_i = abs(f_{org} - \hat{f}_i)$ {Change in surrogate fitness, optimum should have a 0}
  **end if**
  $C \leftarrow C \cup c_i$ {add to list}
**end for**=0
```

A necessary adaptation to this process would involve the expansion of the set of seed solutions generated by any EA such as a GA. One promising line of research would be to generate a set of interpretable surrogate models such as decision trees, each using solutions from differing stages of the EA search trajectory to analyse not just the variable importance created by each but the structural difference between the set of surrogates themselves.

8.4.3 Adaptation to Multi-Objective Trajectories

The techniques developed as part of this thesis have been designed to work with single-objective optimisation problems. There are a significant number of multi-objective evolutionary algorithms available such as NSGA-II [166] which would serve as excellent candidates for the application of our techniques to new problem types. It would be an interesting exercise to fully explore the applicability of our techniques on the search trajectories of these algorithms on multi-objective problems. As noted in Additional Decomposition Methods future work, UMAP represents a possible candidate for further exploration of the effects of decomposition on search trajectories. It may be necessary to incorporate this method when considering the move to multi-objective optimisation problems to help capture the more complex set of interactions in the trajectories, not just between variables but also the trade-offs between objectives.

The approaches necessary to adapt the methods in this thesis to work on multi-objective problems however remains an open question. It is not currently possible to attribute any variable or rankings to any individual objective as the search trajectory structure does not currently include objective values. Future research may include the need to perform analysis on the search trajectories multiple times, with an emphasis on identifying key structures relating to each objective individually and then performing a comparative analysis of the results. This could also include the use of sensitivity analysis or other XAI techniques to measure the effect that observed variance in the data has on a specific objective value rather than overall solution fitness. Work to expand our analysis and trajectory data mining to multi-objective problems could present a fascinating area to explore.

References

[1] Martin Fyvie, John A. W. McCall, Lee A. Christie, Alexandru-Ciprian Zăvoianu, Alexander E. I. Brownlee, and Russell Ainslie. Explaining a Staff Rostering Problem by Mining Trajectory Variance Structures. In *Artificial Intelligence XL: 43rd SGAI International Conference on Artificial Intelligence, AI 2023, Cambridge, UK, December 12–14, 2023, Proceedings*, page 275–290, Berlin, Heidelberg, 2023. Springer-Verlag. doi: https://doi.org/10.1007/978-3-031-47994-6_27.

[2] Martin Fyvie, John A. W. McCall, Lee A. Christie, Alexander E. I. Brownlee, and Manjinder Singh. Towards explainable metaheuristics: Feature extraction from trajectory mining. In *Expert Systems*, page e13494, 2023. doi: https://doi.org/10.1111/exsy.13494.

[3] Martin Fyvie, John A.W. McCall, and Lee A. Christie. "Towards Explainable Metaheuristics: Feature Mining of Search Trajectories through principal component projection". In *SUBMITTED TELO*, pages 89–102. Springer, 2023.

[4] Martin Fyvie, John A. W. McCall, Lee A. Christie, Alexandru-Ciprian Zăvoianu, Alexander E. I. Brownlee, and Russell Ainslie. "Expansion on: Explaining a Staff Rostering Problem by Mining Trajectory Variance Structures". In *SUBMITTED Künstliche Intelligenz (KI) Placeholder*, page x. Springer, 2024.

[5] Martin Fyvie, John A.W. McCall, and Lee A. Christie. Non-deterministic solvers and Explainable AI through Trajectory Mining. pages 75–78. CEUR Workshop Proceedings, 2021.

[6] Martin Fyvie, John A. W. McCall, and Lee A. Christie. Towards Explainable Metaheuristics: PCA for Trajectory Mining in Evolutionary Algorithms. In Max Bramer and Richard Ellis, editors, *Artificial Intelligence XXXVIII*, pages 89–102, Cham, 2021. Springer International Publishing.

[7] 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*, GECCO ’23 Companion, page 1648–1656, New York, NY, USA, 2023. Association for Computing Machinery. doi: https://doi.org/10.1145/3583133.3596353.

[8] Thomas Bäck, Günter Rudolph, and Hans-Paul Schwefel. Evolutionary programming and evolution strategies: Similarities and differences. *Proceedings of the Second Annual Conference on Evolutionary Programming*, 04 1997.

[9] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, 1992.

[10] L. J. Fogel. Autonomous automata, 1962. Industrial Research 4: 14–19.

[11] J. H. Holland. *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press, Ann Arbor, 1975.

[12] I. Rechenberg. *Evolutionsstrategie – Optimierung technisher Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, Stuttgart, GER., 1973.

[13] H. P. Schwefel. *Numerical optimization of computer models.* Wiley Sons, Chichester, 1981.

[14] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[15] Kenneth Price, Rainer M. Storn, and Jouni A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series).* Springer-Verlag, Berlin, Heidelberg, 2005.

[16] Pedro Larranaga and Jose Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation.* 01 2002.

[17] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE transactions on systems, man, and cybernetics, part b (cybernetics)*, 26(1):29–41, 1996.

[18] Wolfgang Banzhaf, Peter Nordin, Robert E Keller, and Frank D Francone. *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications.* Morgan Kaufmann Publishers Inc., 1998.

[19] Octavio Loyola-Gonzalez. Black-box vs. white-box: Understanding their advantages and weaknesses from a practical point of view. *IEEE access*, 7:154096–154113, 2019.

[20] 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.

[21] Milton García-Borroto, José Fco Martínez-Trinidad, and Jesús Ariel Carrasco-Ochoa. Fuzzy emerging patterns for classifying hard domains. *Knowledge and Information Systems*, 28:473–489, 2011.

[22] Octavio Loyola-González, Raúl Monroy, Jorge Rodríguez, Armando López-Cuevas, and Javier Israel Mata-Sánchez. Contrast pattern-based classification for bot detection on twitter. *IEEE Access*, 7:45800–45817, 2019.

[23] Rusul Abduljabbar, Hussein Dia, Sohani Liyanage, and Saeed Asadi Bagloee. Applications of artificial intelligence in transport: An overview. *Sustainability*, 11(1):189, 2019.

[24] Kate Han, Lee A Christie, Alexandru-Ciprian Zăvoianu, and John McCall. On discovering optimal trade-offs when introducing new routes in existing multi-modal public transport systems. In *International Conference on Computer Aided Systems Theory*, pages 104–111. Springer, 2022.

[25] Magnus Eide Schjølberg, Nicklas Paus Bekkevold, Xavier Sánchez-Díaz, and Ole Jakob Mengshoel. Comparing metaheuristic optimization algorithms for ambulance allocation: An experimental simulation study. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1454–1463, 2023.

[26] Gabriela Ochoa, Lee A. Christie, Alexander E. Brownlee, and Andrew Hoyle. Multi-objective evolutionary design of antibiotic treatments. *Artificial Intelligence in Medicine*, 102:101759, 2020.

[27] Alexander EI Brownlee, Jonathan A Wright, Miaomiao He, Timothy Lee, and Paul McMenemy. A novel encoding for separable large-scale multi-objective problems and its application to the optimisation of housing stock improvements. *Applied Soft Computing*, 96:106650, 2020.

[28] Daiki Kiribuchi, Ryoko Hatakeyama, Tomoshi Otsuki, Tatsuya Yoshioka, Kana Konno, and Takumi Matsuda. Combined layout optimization of wind farm and cable connection on complex terrain using a genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1365–1373, 2023.

[29] GDPR Article 13 [Online] Available at:. https://gdpr-info.eu/art-13-gdpr/. [Accessed: 22/02/2024].

[30] GDPR Recital 71 [Online] Available at:. https://gdpr-info.eu/recitals/no-71. [Accessed: 22/02/2024].

[31] European Parliament. EU AI Act p9_ta(2023)0236. https://www.europarl.europa.eu/doceo/document/TA-9-2023-0236_EN.pdf, 2023. Accessed: 10/03/2024.

[32] Balint Gyevnar and Nick Ferguson. Aligning explainable ai and the law: The european perspective. *ArXiv*, abs/2302.10766, 2023.

[33] Tiwonge Msulira Banda, Alexandru-Ciprian Zăvoianu, Andrei Petrovski, Daniel Wöckinger, and Gerd Bramerdorfer. A multi-objective evolutionary approach to discover explainability tradeoffs when using linear regression to effectively model the dynamic thermal behaviour of electrical machines. *ACM Transactions on Evolutionary Learning*, 4(1):1–16, 2024.

[34] Kenneth Sörensen and Fred Glover. Metaheuristics. *Encyclopedia of operations research and management science*, 62:960–970, 2013.

[35] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

[36] Alexandru-Ciprian Zăvoianu, Susanne Saminger-Platz, Edwin Lughofer, and Wolfgang Amrhein. Two enhancements for improving the convergence speed of a robust multi-objective coevolutionary algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 793–800, 2018.

[37] Alexandru-Ciprian Zăvoianu, Benjamin Lacroix, and John McCall. Comparative run-time performance of evolutionary algorithms on multi-objective interpolated continuous optimisation problems. In *Parallel Problem Solving from Nature–PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part I 16*, pages 287–300. Springer, 2020.

[38] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[39] Terry Jones. Evolutionary algorithms, fitness landscapes and search. 06 1995.

[40] Gilbert Strang. Introduction to linear algebra. page 482, 2016.

[41] N.D. Lagaros, M. Kournoutos, N.A. Kallioras, et al. Constraint handling techniques for metaheuristics: a state-of-the-art review and new variants. *Optimization Engineering*, 24:2251–2298, 2023.

[42] Andrej Dobnikar, Nigel C Steele, David W Pearson, Rudolf F Albrecht, Kalyanmoy Deb, and Samir Agrawal. A niched-penalty approach for constraint handling in genetic algorithms. In *Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Portorož, Slovenia, 1999*, pages 235–243. Springer, 1999.

[43] J. Blank and K. Deb. Pymoo: Multi-Objective Optimization in Python. *IEEE Access*, 8:89497–89509, 2020.

[44] Julian Blank. Repair operator - pymoo. 2020. Accessed: 15/02/2024.

[45] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers Operations Research*, 24(11):1097–1100, 1997.

[46] Pierre Hansen, Nenad Mladenović, and Jose A Moreno Perez. Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175:367–407, 2010.

[47] Scott Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science (New York, N.Y.)*, 220:671–80, 06 1983.

[48] Martin Pincus. A monte carlo method for the approximate solution of certain types of constrained optimization problems. *Operations research*, 18(6):1225–1228, 1970.

[49] A Khachaturyan, S Semenovskaya, and B Vainstein. Statistical-thermodynamic approach to determination of structure amplitude phases. *Sov. Phys. Crystallography*, 24(5):519–524, 1979.

[50] A. Khachaturyan, S. Semenovsovskaya, and B. Vainshtein. The thermodynamic approach to the structure analysis of crystals. *Acta Crystallographica Section A*, 37(5):742–754, Sep 1981.

[51] Maximilian A.R. Strobl and Daniel Barker. On simulated annealing phase transitions in phylogeny reconstruction. *Molecular Phylogenetics and Evolution*, 101:46–55, 2016. doi: https://doi.org/10.1016/j.ympev.2016.05.001.

[52] Daniel Barker. Lvb: parsimony and simulated annealing in the search for phylogenetic trees. *Bioinformatics (Oxford, England)*, 20(2):274–275, 2004.

[53] Sean Luke. *Essentials of Metaheuristics*, page 23. Lulu, 2 edition, 2013. ISBN-13 : 78-1300549628.

[54] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers Operations Research*, 13(5):533–549, 1986. Applications of Integer Programming.

[55] Lee A Christie. *The role of Walsh structure and ordinal linkage in the optimisation of pseudo-Boolean functions under monotonicity invariance.* PhD thesis, 2016.

[56] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. volume 1 of *Foundations of Genetic Algorithms*, pages 69–93. Elsevier, 1991.

[57] Alberto Moraglio and Riccardo Poli. Inbreeding properties of geometric crossover and non-geometric recombinations. In *International Workshop on Foundations of Genetic Algorithms*, pages 1–14. Springer, 2007.

[58] Marcel LJ van De Vel. *Theory of convex structures*. Elsevier, 1993.

[59] Alberto Moraglio and Riccardo Poli. Topological interpretation of crossover. In *Genetic and Evolutionary Computation Conference*, pages 1377–1388. Springer, 2004.

[60] Alberto Moraglio and Riccardo Poli. Product geometric crossover. In *International Conference on Parallel Problem Solving from Nature*, pages 1018–1027. Springer, 2006.

[61] Kalyanmoy Deb, Karthik Sindhya, and Tatsuya Okabe. Self-adaptive simulated binary crossover for real-parameter optimization. In *Proceedings of the 9th annual conference on genetic and evolutionary computation*, pages 1187–1194, 2007.

[62] H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions i. binary parameters. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature — PPSN IV*, pages 178–187, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[63] Nix A. Vose M.D. Modeling Genetic Algorithms with Markov Chains. *Mathematics and Artificial Intelligence 5*, 1992.

[64] Siddhartha K. Shakya. *DEUM: a framework for an estimation of distribution algorithm based on Markov random fields.* PhD thesis. COMPLETED.

[65] Jeremy De Bonet, Charles Isbell, and Paul Viola. Mimic: Finding optima by estimating probability densities. *Advances in neural information processing systems*, 9, 1996.

[66] Alexander Edward Ian Brownlee. *Multivariate Markov networks for fitness modelling in an estimation of distribution algorithm.* PhD thesis. COMPLETED.

[67] Martin Pelikan, David E Goldberg, Erick Cantú-Paz, et al. Boa: The bayesian optimization algorithm. In *Proceedings of the genetic and evolutionary computation conference GECCO-99*, volume 1. Citeseer, 1999.

[68] Martin Pelikan, David E Goldberg, and Shigeyoshi Tsutsui. Hierarchical bayesian optimization algorithm: toward a new generation of evolutionary algorithms. In *SICE 2003 annual conference (IEEE Cat. no. 03TH8734)*, volume 3, pages 2738–2743. IEEE, 2003.

[69] Carlos Echegoyen, Jose A Lozano, Roberto Santana, and Pedro Larranaga. Exact bayesian network learning in estimation of distribution algorithms. In *2007 IEEE Congress on Evolutionary Computation*, pages 1051–1058. IEEE, 2007.

[70] Dirk Thierens and Peter AN Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 617–624, 2011.

[71] Bilal, Millie Pant, Hira Zaheer, Laura Garcia-Hernandez, and Ajith Abraham. Differential evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence*, 90:103479, 2020.

[72] Rainer Storn. On the usage of differential evolution for function optimization. In *Proceedings of north american fuzzy information processing*, pages 519–523. Ieee, 1996.

[73] Junhong Liu and Jouni Lampinen. A fuzzy adaptive differential evolution algorithm. *Soft Computing*, 9:448–462, 2005.

[74] Swagatam Das, Ajith Abraham, Uday K Chakraborty, and Amit Konar. Differential evolution using a neighborhood-based mutation operator. *IEEE transactions on evolutionary computation*, 13(3):526–553, 2009.

[75] Yong Wang, Zixing Cai, and Qingfu Zhang. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE transactions on evolutionary computation*, 15(1):55–66, 2011.

[76] Jingqiao Zhang and Arthur C Sanderson. Jade: Self-adaptive differential evolution with fast and reliable convergence performance. In *2007 IEEE congress on evolutionary computation*, pages 2251–2258. IEEE, 2007.

[77] Ryoji Tanabe and Alex Fukunaga. Success-history based parameter adaptation for differential evolution. In *2013 IEEE congress on evolutionary computation*, pages 71–78. IEEE, 2013.

[78] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[79] Zhi-Hui Zhan, Jun Zhang, Yun Li, and Henry Shu-Hung Chung. Adaptive Particle Swarm Optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6):1362–1381, 2009.

[80] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[81] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, jun 2001.

[82] Carlos Coello. Coello, a.c.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. comput. methods appl. mech. engrg. 191(11-12), 1245-1287. *Computer Methods in Applied Mechanics and Engineering*, 191:1245–1287, 01 2002.

[83] Andrea Arcuri and Gordon Fraser. Parameter tuning or default values? an empirical investigation in search-based software engineering. *Empirical Software Engineering*, 18:594–623, 2013.

[84] Xinwei Wang, Alexander E.I. Brownlee, Michal Weiszer, John R. Woodward, Mahdi Mahfouf, and Jun Chen. A chance-constrained programming model for airport ground movement optimisation with taxi time uncertainties. *Transportation Research Part C: Emerging Technologies*, 132:103382, 2021.

[85] Xinwei Wang, Alexander Edward Ian Brownlee, Michal Weiszer, John R. Woodward, Mahdi Mahfouf, and Jun Chen. An interval type-2 fuzzy logic-based map matching algorithm for airport ground movements. *IEEE Transactions on Fuzzy Systems*, 31(2):582–595, 2023.

[86] Joseph Collins, Alexandru-Ciprian Zăvoianu, and John AW McCall. Comparison of simulated annealing and evolution strategies for optimising cyclical rosters with uneven demand and flexible trainee placement. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 451–464. Springer, 2023.

[87] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud '10)*. USENIX Association, 2010.

[88] Carolina Salto, Gabriela Minetti, Enrique Alba, and Gabriel Luque. Big optimization with genetic algorithms: Hadoop, spark, and mpi. *Soft Computing*, pages 1–16, 2023.

[89] Jamal Toutouh and Enrique Alba. A low cost iot cyber-physical system for vehicle and pedestrian tracking in a smart campus. *Sensors*, 22(17):6585, 2022.

[90] Darrell Whitley, Ozeas Quevedo De Carvalho, Mark Roberts, Vivint Shetty, and Piyabutra Jampathom. Scheduling multi-resource satellites using genetic algorithms and permutation based representations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1473–1481, 2023.

[91] Gabriela Ochoa, Lee A Christie, Alexander E Brownlee, and Andrew Hoyle. Multi-objective evolutionary design of antibiotic treatments. *Artificial intelligence in medicine*, 102:101759, 2020.

[92] Mohanad Elhoushy, Belal A Zalam, Amged Sayed, and Essam Nabil. Automated blood glucose regulation for nonlinear model of type-1 diabetic patient under uncertainties: Gwocs type-2 fuzzy approach. *Biomedical Engineering Letters*, pages 1–25, 2023.

[93] Sukhpreet Kaur, Yogesh Kumar, Apeksha Koul, and Sushil Kumar Kamboj. A systematic review on metaheuristic optimization techniques for feature selections in disease diagnosis: open issues and challenges. *Archives of Computational Methods in Engineering*, 30(3):1863–1895, 2023.

[94] Andrei Petrovski and John McCall. Multi-objective optimisation of cancer chemotherapy using evolutionary algorithms. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 531–545. Springer, 2001.

[95] Ziye Song, Xin Guan, and Meng Cheng. Multi-objective optimization strategy for home energy management system including pv and battery energy storage. *Energy Reports*, 8:5396–5411, 2022.

[96] Santosh S Raghuwanshi and Animesh Masih. Renewable energy optimization solutions using meta-heuristics methods. In *Machine Learning and Metaheuristics: Methods and Analysis*, pages 45–72. Springer, 2023.

[97] M. Turek. Explainable artificial intelligence (xai). Funding Program DARPA-BAA-16-53, 2016.

[98] David Gunning and David W. Aha. Darpa's explainable artificial intelligence program. *AI Magazine*, 40(2):44–58, 2019.

[99] Amina Adadi and Mohammed Berrada. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, 6:52138–52160, 2018.

[100] Dignum, Virginia. The ART of AI - Accountability, Responsibility, Transparency. *Medium*, Mar 2018.

[101] The PHG Foundation. [Online] Available at:. http://www.phgfoundation.org/. [Accessed: 21/09/2023].

[102] Information Commissioners Office. [Online] Available at:. https://ico.org.uk. [Accessed: 18/01/2023].

[103] The Alan Turing Institute [Online] Available at:. https://www.turing.ac.uk/, Jan 2021. [Accessed: 18/01/2024].

[104] Johan Ordish, Tanya Brigden, and Alison Hall. Black box medicine and transparency | PHG Foundation. page 34, 2020.

[105] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence*, 267:1–38, 2019.

[106] Wesley C. Salmon. *Four Decades of Scientific Explanation*. University of Pittsburgh Press, 2006.

[107] James Woodward. *Making Things Happen: A Theory of Causal Explanation*. Oxford University Press, 2005.

[108] Tania Lombrozo. Causal–explanatory pluralism: How intentions, functions, and mechanisms influence causal ascriptions. *Cognitive Psychology*, 61(4):303–332, 2010.

[109] Joseph Y. Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part ii: Explanations. *The British Journal for the Philosophy of Science*, 56(4):889–911, 2005.

[110] C. Ginet. In defense of a non-causal account of reasons explanations. *Journal of Ethics*, 12:229–237, 2008.

[111] Alexander Wendt. On constitution and causation in international relations. *Review of International Studies*, 24(5):101–118, 1998.

[112] Zachary C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery, 2017.

[113] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

[114] Ryan Zhou and Ting Hu. Evolutionary approaches to explainable machine learning. *arXiv preprint arXiv:2306.14786*, 2023.

[115] 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.

[116] Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

[117] 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*, pages 1135–1144, 2016.

[118] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[119] Robert Hoffman, Shane Mueller, Gary Klein, Mohammadreza Jalaeian, and Connor Tate. Explainable ai: roles and stakeholders, desirements and challenges. *Frontiers in Computer Science*, 5, 08 2023.

[120] Alun Preece, Dan Harborne, Dave Braines, Richard Tomsett, and Supriyo Chakraborty. Stakeholders in explainable ai. *arXiv preprint arXiv:1810.00184*, 2018.

[121] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. *arXiv preprint arXiv:1909.03012*, 2019.

[122] Shruthi Chari, Oshani Seneviratne, Daniel M Gruen, Morgan A Foreman, Amar K Das, and Deborah L McGuinness. Explanation ontology: a model of explanations for user-centered ai. In *International Semantic Web Conference*, pages 228–243. Springer, 2020.

[123] Sina Mohseni, Niloofar Zarei, and Eric D Ragan. A multidisciplinary survey and framework for design and evaluation of explainable ai systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 11(3-4):1–45, 2021.

[124] 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), 2023.

[125] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning, 2017.

[126] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael A. Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 80–89, 2018.

[127] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, 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.

[128] Jean-Marc Fellous, Guillermo Sapiro, Andrew Rossi, Helen Mayberg, and Michele Ferrante. Explainable artificial intelligence for neuroscience: Behavioral neurostimulation. *Frontiers in Neuroscience*, 13, 2019.

[129] B Khaleghi. The how of explainable ai: pre-modelling explainability. *URL: https://towardsdatascience. com/the-how-of-explainable-ai-pre-modellingexplainability-699150495fe4*, 2019.

[130] Jaume Bacardit, Alexander EI Brownlee, Stefano Cagnoni, Giovanni Iacca, John McCall, and David Walker. The intersection of evolutionary computation and explainable ai. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1757–1762, 2022.

[131] Kalyanmoy Deb, Sunith Bandaru, David Greiner, António Gaspar-Cunha, and Cem Celal Tutum. An integrated approach to automated innovization for discovering useful design principles: Case studies from engineering. *Applied Soft Computing*, 15:42–56, 2014.

[132] Kalyanmoy Deb and Aravind Srinivasan. Innovization: Discovery of innovative design principles through multiobjective evolutionary optimization. *Multiobjective Problem Solving from Nature: From Concepts to Applications*, pages 243–262, 2008.

[133] Karl Pearson. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, November 1901.

[134] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[135] S. M.Holland. Principal components analysis (pca). 2019. [Online] Available at chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://stratigrafia.org/8370/handouts/pcaTutorial.pdf [Accessed: 01/03/2024].

[136] Geoffrey E. Hinton and Sam T. Roweis. Stochastic neighbor embedding. In *Neural Information Processing Systems*, 2002.

[137] Mujtaba Husnain, Malik Muhammad Saad Missen, Shahzad Mumtaz, Dost Muhammad Khan, Mickäel Coustaty, Muhammad Muzzamil Luqman, Jean-Marc Ogier, Hizbullah Khattak, Sikandar Ali, Ali Samad, and Shahzad Sarfraz. Urdu handwritten characters data visualization and recognition using distributed stochastic neighborhood embedding and deep network. *Complex.*, 2021, jan 2021.

[138] Saurav Jadhav. What is t-SNE? Analytics Vidhya, Jun 2021. Available [Online] at https://medium.com/analytics-vidhya/what-is-t-sne-37bfb920e431.

[139] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(nov):2579–2605, 2008.

[140] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[141] Rui Huang, Qingshan Liu, Hanqing Lu, and Songde Ma. Solving the small sample size problem of lda. In *2002 International Conference on Pattern Recognition*, volume 3, pages 29–32 vol.3, 2002.

[142] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[143] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[144] Scikit learn contributors. Partial dependence plots in scikit-learn. 2024. Available [Online] at https://scikit-learn.org/stable/auto_examples/inspection/plot_partial_dependence.html.

[145] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

[146] S. Wright. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In *Proceedings of the Sixth International Congress on Genetics*, pages 356–366, Ithaca, NY, USA.

[147] Katherine M Malan and Andries P Engelbrecht. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163, 2013.

[148] Katherine Mary Malan. A survey of advances in landscape analysis for optimisation. *Algorithms*, 14(2), 2021.

[149] Hendrik Richter. Coupled map lattices as spatio-temporal fitness functions: Landscape measures and evolutionary optimization. *Physica D: Nonlinear Phenomena*, 237(2):167–186, 2008.

[150] Jean-Paul Watson. An introduction to fitness landscape analysis and cost models for local search. *Handbook of metaheuristics*, pages 599–623, 2010.

[151] Arnaud Liefooghe, Sébastien Verel, Hernán Aguirre, and Kiyoshi Tanaka. What makes an instance difficult for black-box 0–1 evolutionary multiobjective optimizers? In *International Conference on Artificial Evolution (Evolution Artificielle)*, pages 3–15. Springer, 2013.

[152] Fabio Daolio, Arnaud Liefooghe, Sébastien Verel, Hernán Aguirre, and Kiyoshi Tanaka. Global vs local search on multi-objective nk-landscapes: contrasting the impact of problem features. In *proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 369–376, 2015.

[153] Gabriela Ochoa, Katherine Malan, and Christian Blum. Search Trajectory Networks: A Tool for Analysing and Visualising the Behaviour of Metaheuristics. *Applied Soft Computing*, 109:107492, 05 2021.

[154] Jason Adair, Gabriela Ochoa, and Katherine M. Malan. Local optima networks for continuous fitness landscapes. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 1407–1414, New York, NY, USA, 2019. Association for Computing Machinery.

[155] Gabriela Ochoa, Katherine Malan, and Christian Blum. Search Trajectories Illuminated. *ACM SIGEVOlution*, 14:1–5, 07 2021.

[156] Arnaud Liefooghe, Gabriela Ochoa, Sébastien Verel, and Bilel Derbel. Pareto local optimal solutions networks with compression, enhanced visualization and expressiveness. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 713–721, New York, NY, USA, 2023. Association for Computing Machinery.

[157] Risto Trajanov, Stefan Dimeski, Martin Popovski, Peter Korošec, and Tome Eftimov. Explainable landscape-aware optimization performance prediction, 2021.

[158] Risto Trajanov, Stefan Dimeski, Martin Popovski, Peter Korošec, and Tome Eftimov. Explainable landscape analysis in automated algorithm performance prediction, 2022.

[159] 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*, pages 1135–1144. ACM, 2016.

[160] SHAP Contributors. An introduction to Explainable AI with Shapley Values. 2023. Available [Online] at https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html.

[161] Philip E Tetlock and Aaron Belkin. *Counterfactual thought experiments in world politics: Logical, methodological, and psychological perspectives*. Princeton University Press, 1996.

[162] Jack S Levy. Counterfactuals, causal inference, and historical analysis. *Security Studies*, 24(3):378–402, 2015.

[163] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr, 2018.

[164] Susanne Dandl, Christoph Molnar, Martin Binder, and Bernd Bischl. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*, pages 448–469. Springer, 2020.

[165] J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27(4):857, December 1971.

[166] 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.

[167] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, 2018.

[168] Alberto Fernández, Francisco Herrera, Oscar Cordón, María José del Jesus, and Francesco Marcelloni. Evolutionary fuzzy systems for explainable artificial intelligence: Why, when, what for, and where to? *IEEE Computational Intelligence Magazine*, 14(1):69–81, 2019.

[169] Jaume Bacardit, Edmund K Burke, and Natalio Krasnogor. Improving the scalability of rule-based evolutionary learning. *Memetic computing*, 1:55–67, 2009.

[170] Romaissaa Mazouni and Abdellatif Rahmoun. Agge: A novel method to automatically generate rule induction classifiers using grammatical evolution. In *Intelligent Distributed Computing VIII*, pages 279–288. Springer, 2015.

[171] Yashesh Dhebar, Kalyanmoy Deb, Subramanya Nageshrao, Ling Zhu, and Dimitar Filev. Interpretable-ai policies using evolutionary nonlinear decision trees for discrete action systems. *arXiv preprint arXiv:2009.09521*, 2020.

[172] Xavier Llorà and Stewart W. Wilson. Mixed decision trees: Minimizing knowledge representation bias in lcs. In Kalyanmoy Deb, editor, *Genetic and Evolutionary Computation – GECCO 2004*, pages 797–809, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[173] Marta Galende, G Sainz, and Maria J Fuente. Accuracy-interpretability balancing in fuzzy models based on multiobjective genetic algorithm. In *2009 European Control Conference (ECC)*, pages 3915–3920. IEEE, 2009.

[174] Alon Jacovi. Trends in explainable ai (xai) literature. *arXiv preprint arXiv:2301.05433*, 2023.

[175] John Debs. Thefuzz. https://github.com/seatgeek/thefuzz, 2023.

[176] David E Goldberg. Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction. *Complex Systems*, 3:129–152, 1989.

[177] Darrell Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2), 1994. doi: 10.1007/bf00175354.

[178] Laurens Van Der Maaten, Eric Postma, Jaap Van den Herik, et al. Dimensionality reduction: a comparative review. *J Mach Learn Res*, 10(66-71):13, 2009.

[179] Kilian Q Weinberger, Fei Sha, and Lawrence K Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the twenty-first international conference on Machine learning*, page 106, 2004.

[180] Geoffrey E Hinton and Sam Roweis. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15, 2002.

[181] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[182] Ian T Jolliffe. Principal component analysis. volume 45, page 276. American Society for Quality, 2003.

[183] Giorgia Pasini. Principal component analysis for stock portfolio management. *International Journal of Pure and Applied Mathematics*, 115(1):153–167, 2017.

[184] Fangzhou Yao, Jeff Coquery, and Kim-Anh Lê Cao. Independent principal component analysis for biologically meaningful dimension reduction of large biological data sets. *BMC bioinformatics*, 13:1–15, 2012.

[185] Matheus Pereira Libório, Oseias da Silva Martinuci, Alexei Manso Correa Machado, Thiago Melo Machado-Coelho, Sandro Laudares, and Patrícia Bernardes. Principal component analysis applied to multidimensional social indicators longitudinal studies: limitations and possibilities. *GeoJournal*, 87(3):1453–1468, 2022.

[186] Michael Greenacre. *Correspondence Analysis in Practice*. Chapman and Hall/CRC, 2007.

[187] Fabio Manca, Angela Maria D'Uggento, and Nicola Convertini. Customer segmentation through multiple correspondence analysis. In *2018 AEIT International Annual Conference*, pages 1–5, 2018.

[188] Johs Hjellbrekke. *Multiple Correspondence Analysis For The Social Sciences*. 06 2018.

[189] Wentian Li, Jane E Cerise, Yaning Yang, and Henry Han. Application of t-sne to human genetic data. *Journal of bioinformatics and computational biology*, 15(04):1750017, 2017.

[190] Ujang Riswanto. Improving image analysis with t-sne and blind visualization techniques. 2023. Available [Online] at https://ujangriswanto08.medium.com/improving-image-analysis-with-t-sne-and-blind-visualization-techniques-7a6cf3a18bbc.

[191] Alan J. Izenman. Linear discriminant analysis. Springer, 2013.

[192] Jason Brownlee. Linear discriminant analysis for dimensionality reduction in python. *Machine Learning Mastery*, 2020. Available [Online] at http://machinelearningmastery.com/linear-discriminant-analysis-for-dimensionality-reduction-in-python/.

[193] Fatma Zohra Chelali, A Djeradi, and R Djeradi. Linear discriminant analysis for face recognition. In *2009 International Conference on Multimedia Computing and Systems*, pages 1–10. IEEE, 2009.

[194] I. de Zarzà, J. de Curtò, and Carlos T. Calafate. Umap for geospatial data visualization. *Procedia Computer Science*, 225:1661–1671, 2023. 27th International Conference on Knowledge Based and Intelligent Information and Engineering Sytems (KES 2023).

[195] Alex Diaz-Papkovich, Luke Anderson-Trocmé, Chief Ben-Eghan, and Simon Gravel. Umap reveals cryptic population structure and phenotype heterogeneity in large genomic cohorts. *PLoS genetics*, 15(11):e1008432, 2019.

[196] Nan Xu, Yuxiang Zhou, Ameet Patel, Na Zhang, and Yongming Liu. Parkinson's disease diagnosis beyond clinical features: A bio-marker using topological machine learning of resting-state functional magnetic resonance imaging. *Neuroscience*, 509:43–50, 2023.

[197] Geoffrey E. Hinton and Ruslan R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):1004–1007, 2006.

[198] Mahmood Yousefi-Azar, Vijay Varadharajan, Len Hamey, and Uday Tupakula. Autoencoder-based feature learning for cyber security applications. In *2017 International joint conference on neural networks (IJCNN)*, pages 3854–3861. IEEE, 2017.

[199] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 665–674, 2017.

[200] Jin Zheng and Lihui Peng. An autoencoder-based image reconstruction for electrical capacitance tomography. *IEEE Sensors Journal*, 18(13):5464–5474, 2018.

[201] Larry R. Fabrigar and Douglas T. Wegener. *Exploratory Factor Analysis*. Oxford University Press, 2011.

[202] ASMM Hoque and Zainudin Awang. Exploratory factor analysis of entrepreneurial marketing: Scale development and validation in the sme context of bangladesh. In *Proceedings of the International Social Sciences and Tourism Research Conference*, pages 22–38, 2016.

[203] Noora Shrestha. Factor analysis as a tool for survey analysis. *American Journal of Applied Mathematics and Statistics*, 9(1):4–11, 2021.

[204] Joshua B. Tenenbaum, Vinay de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[205] Ashutosh Saxena, Abhinav Gupta, and Amitabha Mukerjee. Non-linear dimensionality reduction by locally linear isomaps. In *International Conference on Neural Information Processing*, pages 1038–1043. Springer, 2004.

[206] Ameet Talwalkar, Sanjiv Kumar, and Henry Rowley. Large-scale manifold learning. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[207] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–793, 1999.

[208] Paresh Chandra Barman, Nadeem Iqbal, and Soo-Young Lee. Non-negative matrix factorization based text mining: feature extraction and classification. In *International Conference on Neural Information Processing*, pages 703–712. Springer, 2006.

[209] David Guillamet, Jordi Vitria, and Bernt Schiele. Introducing a weighted non-negative matrix factorization for image classification. *Pattern Recognition Letters*, 24(14):2447–2454, 2003.

[210] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis. *Neural computation*, 21(3):793–830, 2009.

[211] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[212] Zhengqin Li and Jiansheng Chen. Superpixel segmentation using linear spectral clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1356–1363, 2015.

[213] Francis R Bach and Michael I Jordan. Learning spectral clustering, with application to speech separation. *The Journal of Machine Learning Research*, 7:1963–2001, 2006.

[214] Desmond J Higham, Gabriela Kalna, and Milla Kibble. Spectral clustering and its use in bioinformatics. *Journal of computational and applied mathematics*, 204(1):25–37, 2007.

[215] John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research*, 16(1):2859–2900, 2015.

[216] Alexander EI Brownlee, Martin Pelikan, John AW McCall, and Andrei Petrovski. An application of a multivariate estimation of distribution algorithm to cancer chemotherapy. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 463–464, 2008.

[217] Shumeet Baluja, Scott Davies, et al. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. Carnegie-Mellon University. Department of Computer Science, 1997.

[218] Stephanie Forrest and Melanie Mitchell. Relative building-block fitness and the building-block hypothesis. In L. DARRELL WHITLEY, editor, *Foundations of Genetic Algorithms*, volume 2 of *Foundations of Genetic Algorithms*, pages 109–126. Elsevier, 1993.

[219] Melanie Mitchell, Stephanie Forrest, and John H. Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. 1991.

[220] David E. Goldberg. Genetic algorithms and walsh functions: Part i, a gentle introduction. *Complex Syst.*, 3, 1989.

[221] David E. Goldberg. Genetic algorithms and walsh functions: Part ii, deception and its analysis. *Complex Syst.*, 3, 1989.

[222] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. Real-parameter black-box optimization benchmarking 2010: Presentation of the noiseless functions. 2010.

[223] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[224] Vincenzo Cutello, Giuseppe Nicosia, Mario Pavone, and Giovanni Stracquadanio. Entropic divergence for population based optimization algorithms. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2010. doi: 10.1109/CEC.2010.5586044.

[225] David JC MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[226] Kevin G. Dunn. Process improvement using data. Zenodo, June 2023. Available [Online] at https://doi.org/10.5281/zenodo.8021769.

[227] Jérôme Pagès. *Multiple Factor Analysis by Example Using R*. CRC Press, 1 edition, 2014.

[228] William Webber, Alistair Moffat, and Justin Zobel. A Similarity Measure for Indefinite Rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4):1–38, 2010.

[229] Charles Spearman. The proof and measurement of association between two things. Appleton-Century-Crofts, 1961.

[230] M. G. Kendall. A New Measure of Rank Correlation. volume 30, pages 81–93. Biometrika, 06 1938.

[231] Krupesh Raikar. How to Objectively Compare Two Ranked Lists in Python, 2023. [online] https://towardsdatascience.com/how-to-objectively-compare-two-ranked-lists-in-python-b3d74e236f6a.

[232] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. *Proceedings of the 9th Python in Science Conference*, 2010, 01 2010.

[233] Nguyen Thi Hien and Nguyen Xuan Hoai. A brief overview of population diversity measures in genetic programming. In *Proc. 3rd Asian-Pacific Workshop on Genetic Programming, Hanoi, Vietnam*, pages 128–139, 2006.

[234] Ronald W Morrison and Kenneth A De Jong. Measurement of population diversity. In *International conference on artificial evolution (evolution artificielle)*, pages 31–41. Springer, 2001.

[235] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[236] Shumeet Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical report, USA, 1994.

[237] Shumeet Baluja. An empirical comparison of seven iterative and evolutionary heuristics for static function optimization. In *Proc. of the 11th Int. Conf. on System Engineering*, pages 692–697. Citeseer, 1995.

[238] Kalyanmoy Deb and Debayan Deb. Analysing Mutation Schemes for Real-Parameter Genetic Algorithms. *International Journal of Artificial Intelligence and Soft Computing*, 4:1–28, 2014.

[239] Kalyanmoy Deb, Karthik Sindhya, and Tatsuya Okabe. Self-Adaptive Simulated Binary Crossover for Real-Parameter Optimization. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, page 1187–1194, New York, NY, USA, 2007. Association for Computing Machinery.

[240] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[241] P Valkovič. BBOB Torch. https://github.com/PatrikValkovic/BBOBtorch, 2021.

[242] Hidefumi Katsuura. Continuous nowhere-differentiable functions—an application of contraction mappings. *The American Mathematical Monthly*, 98(5):411–416, 1991.

[243] Radka Poláková, Josef Tvrdík, and Petr Bujok. Differential evolution with adaptive mechanism of population size according to current population diversity. *Swarm and Evolutionary Computation*, 50:100519, 2019.

[244] Adam P. Piotrowski, Jaroslaw J. Napiorkowski, and Agnieszka E. Piotrowska. Population size in particle swarm optimization. *Swarm and Evolutionary Computation*, 58:100718, 2020.

[245] 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. Springer International Publishing, 2018.

[246] Ellen R Girden. *ANOVA: Repeated measures*. Number 84. Sage, 1992.

[247] Kenneth N. Reid, Jingpeng Li, Alexander Brownlee, Mathias Kern, Nadarajen Veerapen, Jerry Swan, and Gilbert Owusu. A Hybrid Metaheuristic Approach to a Real World Employee Scheduling Problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, page 1311–1318, New York, NY, USA, 2019. Association for Computing Machinery.

[248] Martin Fyvie, John A.W. McCall, Lee A. Christie, Alexandru-Ciprian Zăvoianu, Alexander E.I. Brownlee, and Russell Ainslie. Explaining A Staff Rostering Problem By Mining Trajectory Variance Structures Definition. https://github.com/rgu-subsea/mfyvie_sgai2023_varstruct.git, 2023.

[249] Olive Jean Dunn. Multiple Comparisons Among Means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[250] Yoav Benjamini and Yosef Hochberg. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995.

[251] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm Evol. Comput.*, 1(2):61–70, 2011.

[252] A. E. I. Brownlee, J.R. Woodward, and J. Swan. Metaheuristic design pattern: Surrogate fitness functions. In *MetaDeeP Workshop, Proc. GECCO Companion*, pages 1261–1264, Madrid, Spain, 2015. ACM Press.

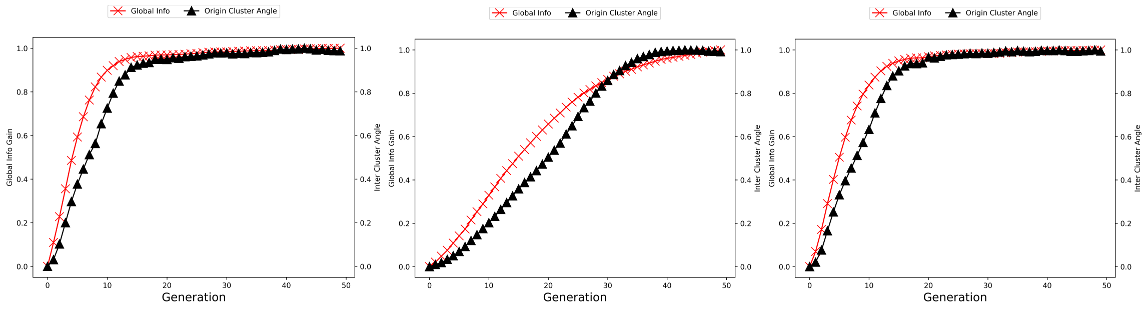
[253] Aidan Wallace, Alexander E. I. Brownlee, and David Cairns. Towards Explaining Metaheuristic: Solution Quality by Data Mining Surrogate Fitness Models for Importance of Variables. In Max Bramer and Richard Ellis, editors, *Artificial Intelligence XXXVIII*, pages 58–72, Cham, 2021. Springer International Publishing.

Appendix A

Binary Problem Information Gain Results

A.1 RoyalRoad

A.1.1 Information Gain - Global

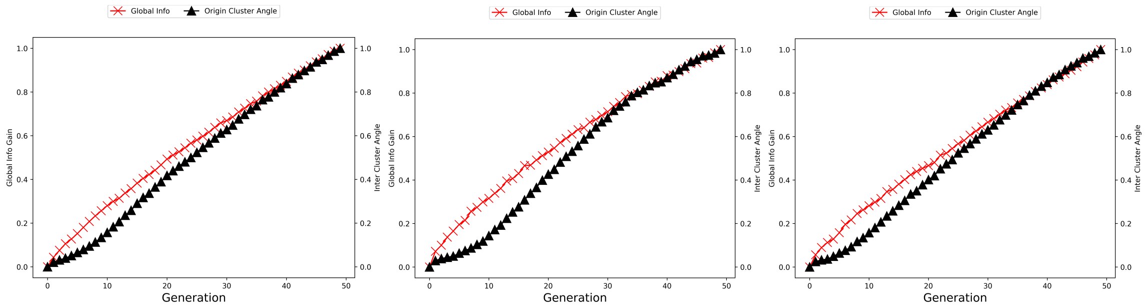


(a) GA Royal Road Trunc20 Mean Global

(b) GA Royal Road Trunc50 Mean Global

(c) GA Royal Road Tour Mean Global

Fig. A.1 GA Global Information Vs PCA on Royal Road by Selection



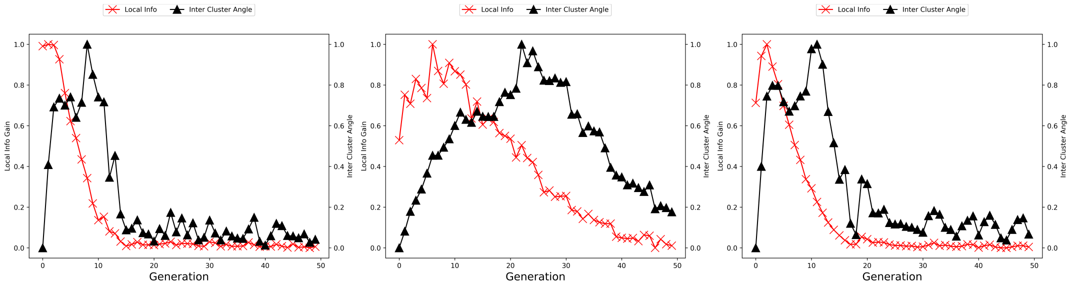
(a) PBIL Royal Road Trunc20 Mean Global

(b) PBIL Royal Road Trunc50 Mean Global

(c) PBIL Royal Road Tour Mean Global

Fig. A.2 PBIL Global Information Vs PCA Angles on Royal Road by Selection

A.1.2 Information Gain - Local

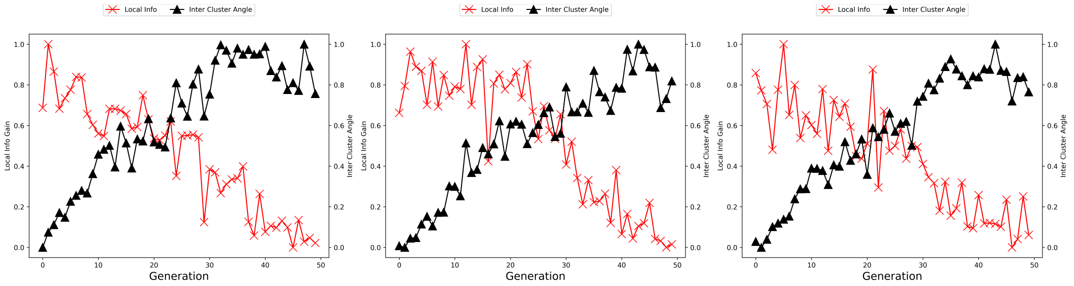


(a) GA Royal Road Trunc20 Mean Local

(b) GA Royal Road Trunc50 Mean Local

(c) GA Royal Road Tour Mean Local

Fig. A.3 GA Local Information Vs PCA Angles on Royal Road by Selection



(a) PBIL Royal Road Trunc20 Mean Local

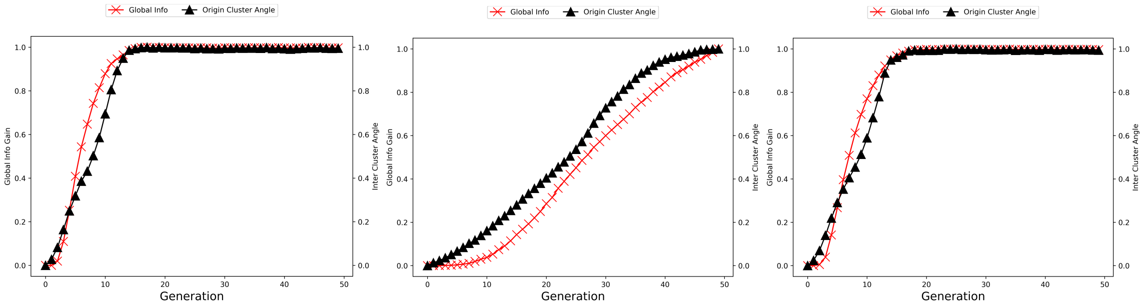
(b) PBIL Royal Road Trunc50 Mean Local

(c) PBIL Royal Road Tour Mean Local

Fig. A.4 PBIL Local Information Vs PCA Angles on Royal Road by Selection

A.2 Trap5

A.2.1 Information Gain - Global

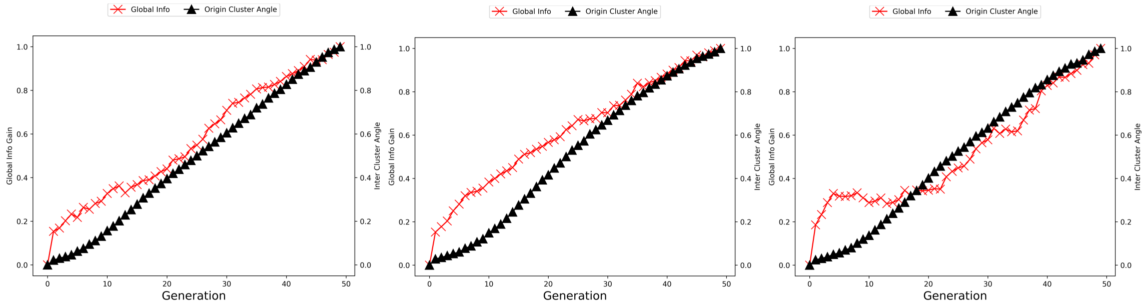


(a) GA Trap5 Trunc20 Mean Global

(b) GA Trap5 Trunc50 Mean Global

(c) GA Trap5 Tour Mean Global

Fig. A.5 GA Global Information Vs PCA Angles on Trap5 by Selection



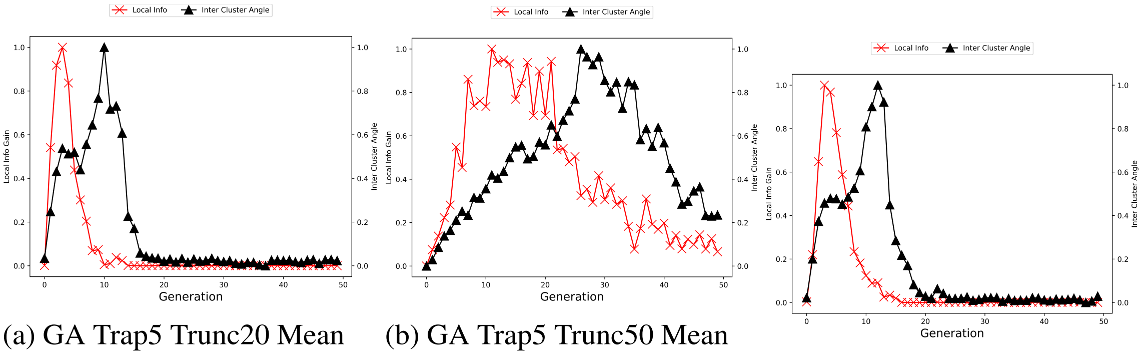
(a) PBIL Trap5 Trunc20 Mean Global

(b) PBIL Trap5 Trunc50 Mean Global

(c) PBIL Trap5 Tour Mean Global

Fig. A.6 PBIL Global Information Vs PCA Angles on Trap5 by Selection

A.2.2 Information Gain - Local

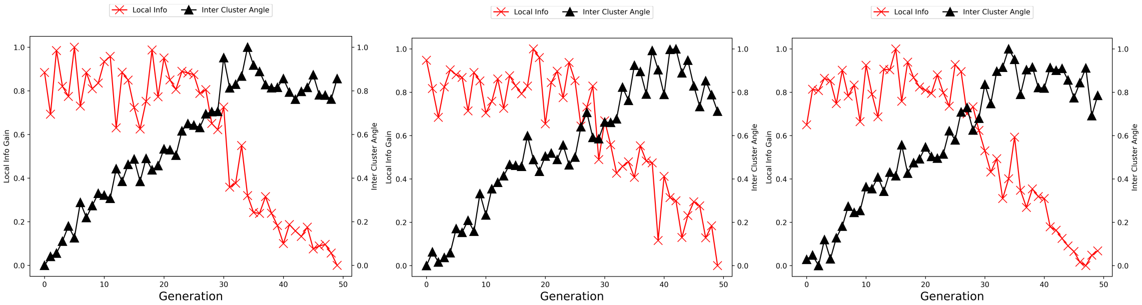


(a) GA Trap5 Trunc20 Mean Local

(b) GA Trap5 Trunc50 Mean Local

(c) GA Trap5 Tour Mean Local

Fig. A.7 GA Local Information Vs PCA Angles on Trap5 by Selection



(a) PBIL Trap5 Trunc20 Mean Local

(b) PBIL Trap5 Trunc50 Mean Local

(c) PBIL Trap5 Tour Mean Local

Fig. A.8 PBIL Local Information Vs PCA Angles on Trap5 by Selection

Appendix B

Variable Contribution and Alignment in the BBOB Problems Additional Tables and Figures

B.1 Algorithm Performance All BBOB

Alg	Prob	F-Opt	Median Best F.	Min	Max	Std	Dist. To. Opt
CMAES	F1	129.88	129.8837	129.8837	129.8837	0	0.003698
DE	F1	129.88	129.8837	129.8837	129.8837	0	0.003698
GA	F1	129.88	129.8837	129.8837	129.8839	2.26E-05	0.003728
PSO	F1	129.88	129.8837	129.8837	129.8837	0	0.003698
CMAES	F2	129.88	129.8837	129.8837	129.8837	0	0.003698
DE	F2	129.88	129.8837	129.8837	129.8837	0	0.003698
GA	F2	129.88	130.4343	129.9056	135.0698	1.152305	0.554326
PSO	F2	129.88	129.8837	129.8837	129.8837	1.52E-06	0.003698
CMAES	F3	129.88	137.7932	130.8724	152.6234	3.475602	7.913182
DE	F3	129.88	153.4348	145.2051	162.9933	4.142318	23.55484
GA	F3	129.88	129.8947	129.8846	129.9456	0.012071	0.014684
PSO	F3	129.88	132.9703	129.8837	145.644	2.896078	3.090268
CMAES	F4	129.88	142.7366	136.8045	161.5216	4.13893	12.8566
DE	F4	129.88	155.0278	142.8064	162.0462	4.500036	25.1478
GA	F4	129.88	129.9028	129.8849	130.3522	0.047803	0.022763
PSO	F4	129.88	136.0396	130.8795	150.8246	3.637451	6.159619
CMAES	F5	129.88	129.8837	129.8837	129.8837	0	0.003698

DE	F5	129.88	129.8837	129.8837	129.8837	2.14E-06	0.003698
GA	F5	129.88	129.8837	129.8837	129.8837	2.99E-06	0.003698
PSO	F5	129.88	129.8837	129.8837	129.8837	0	0.003698
CMAES	F6	129.88	129.8837	129.8837	129.8837	7.44E-06	0.003698
DE	F6	129.88	129.889	129.8852	130.0019	0.011724	0.009008
GA	F6	129.88	130.2448	129.9608	131.4424	0.271382	0.364835
PSO	F6	129.88	129.8924	129.8856	129.9176	0.005823	0.01241
CMAES	F7	129.88	129.8837	129.8837	131.0331	0.355521	0.003698
DE	F7	129.88	129.8837	129.8837	130.6674	0.110208	0.003698
GA	F7	129.88	132.9133	130.4335	144.263	2.540618	3.033254
PSO	F7	129.88	130.99	129.8879	136.5948	1.008478	1.109967
CMAES	F8	129.88	129.8837	129.8837	133.8703	0.39666	0.003698
DE	F8	129.88	134.0839	131.2206	136.1108	1.173451	4.203908
GA	F8	129.88	134.8879	129.9524	191.6465	6.356945	5.007917
PSO	F8	129.88	135.3384	129.9116	195.0592	8.191905	5.458402
CMAES	F9	1000	1000	1000	1000	6.07E-06	0
DE	F9	1000	1005.057	1001.621	1008.68	1.179986	5.057129
GA	F9	1000	1008.715	1000.276	1254.087	48.14198	8.715179
PSO	F9	1000	1007.445	1000.209	1156.339	24.00853	7.44458
CMAES	F10	1000	1000	1000	1006.425	0.81086	0
DE	F10	1000	1004.051	1000.087	1113.835	24.92777	4.050751
GA	F10	1000	4407.775	1279.581	22050.3	3793.574	3407.775
PSO	F10	1000	2059.042	1037.8	6282.497	1120.708	1059.042
CMAES	F11	1000	1000	1000	1000	0	0
DE	F11	1000	1000.475	1000.003	1010.486	1.370566	0.475372
GA	F11	1000	1060.202	1008.895	1133.103	28.44615	60.20239
PSO	F11	1000	1030.01	1004.571	1132.26	22.15788	30.01031
CMAES	F12	1000	1000	1000	1006.39	1.282499	0.000183
DE	F12	1000	1000.514	1000.093	1005.503	1.133905	0.51358
GA	F12	1000	1016.717	1001.443	1261.902	28.04532	16.71741
PSO	F12	1000	1001.153	1000	1010.331	2.529329	1.152771
CMAES	F13	1000	1000	1000	1000	2.77E-05	6.1E-05
DE	F13	1000	1000.105	1000.039	1000.283	0.041767	0.104858
GA	F13	1000	1004.044	1001.213	1036.922	7.903583	4.043945
PSO	F13	1000	1003.519	1000.001	1033.089	8.718286	3.519226
CMAES	F14	1000	1000	1000	1000	1.45E-05	0

DE	F14	1000	1000	1000	1000	3.90E-05	0
GA	F14	1000	1000.007	1000.002	1000.013	0.002261	0.007324
PSO	F14	1000	1000.001	1000	1000.002	0.000234	0.00061
CMAES	F15	1000	1006.921	1000	1024.932	4.739127	6.920837
DE	F15	1000	1035.727	1022.694	1046.386	5.215436	35.72729
GA	F15	1000	1025.017	1006.021	1055.375	10.87465	25.01678
PSO	F15	1000	1020.762	1004.943	1064.264	11.90002	20.76242
CMAES	F16	1000	999.9854	999.9854	1000.082	0.014365	-0.01459
DE	F16	1000	1007.243	1003.253	1010.741	1.399405	7.242889
GA	F16	1000	1001.883	1000.055	1006.434	1.58869	1.883301
PSO	F16	1000	1000.363	999.9939	1007.24	1.084992	0.363434
CMAES	F17	1000	1000	1000	1000.019	0.002053	6.1E-05
DE	F17	1000	1000.001	1000	1000.004	0.000873	0.001434
GA	F17	1000	1000.191	1000.024	1000.689	0.153522	0.190582
PSO	F17	1000	1000.086	1000.001	1000.654	0.129891	0.086365
CMAES	F18	1000	1000	1000	1000.351	0.041823	6.1E-05
DE	F18	1000	1000.056	1000.019	1000.247	0.033406	0.056488
GA	F18	1000	1002.991	1000.292	1007.859	1.774557	2.991119
PSO	F18	1000	1000.932	1000.059	1009.826	1.686224	0.932098
CMAES	F19	1000	1000.222	1000.036	1000.98	0.207437	0.221802
DE	F19	1000	1002.796	1001.13	1003.868	0.55593	2.796173
GA	F19	1000	1001.824	1000.371	1004.353	0.75435	1.823792
PSO	F19	1000	1002.204	1000.677	1003.85	0.576471	2.20401
CMAES	F20	129.88	131.3445	130.4562	132.0947	0.369113	1.464482
DE	F20	129.88	131.7687	130.6505	132.19	0.235678	1.888677
GA	F20	129.88	130.463	130.0037	131.2665	0.238719	0.583028
PSO	F20	129.88	130.6733	130.0399	131.2837	0.267404	0.793294
CMAES	F21	62.21	63.61864	63.61864	63.61864	0	1.408641
DE	F21	62.21	63.42977	62.21315	68.02241	0.952167	1.219771
GA	F21	62.21	63.61866	62.21315	69.20274	1.983568	1.408664
PSO	F21	62.21	63.66929	62.21315	85.87863	3.658475	1.459289
CMAES	F22	1000	1020.562	1002.523	1052.954	9.695194	20.56226
DE	F22	1000	1004.281	1002.523	1007.159	1.3931	4.281067
GA	F22	1000	1005.829	1002.528	1027.438	5.790258	5.82901
PSO	F22	1000	1005.828	1002.523	1059.25	10.30779	5.828369
CMAES	F23	129.88	129.9195	129.8837	131.6227	0.194818	0.039548

DE	F23	129.88	131.4158	130.8393	132.0701	0.264654	1.535779
GA	F23	129.88	130.8568	130.2254	131.9019	0.370717	0.976804
PSO	F23	129.88	131.4048	130.4849	132.1611	0.359095	1.524785
CMAES	F24	-0.2	14.9819	1.667411	35.88194	4.668089	15.1819
DE	F24	-0.2	46.46808	32.42996	55.94858	5.113254	46.66808
GA	F24	-0.2	35.6216	14.75552	65.87791	10.16499	35.8216
PSO	F24	-0.2	36.54667	16.01192	63.09313	9.83923	36.74667

Table B.1 Full BBOB Algorithm Performance Results

B.2 Variable Rankings Across 3PCs - All BBOB

Alg	Prob	Top 4 PC1	Top 4 PC2	Top 4 PC3
CMAES	F1	3, 4, 8, 5	7, 9, 10, 6	6, 9, 10, 1
DE	F1	3, 8, 4, 2	5, 2, 4, 3	2, 5, 8, 10
GA	F1	3, 8, 4, 2	2, 4, 5, 3	5, 2, 3, 10
PSO	F1	3, 4, 8, 5	2, 9, 5, 7	1, 9, 7, 5
CMAES	F2	8, 4, 5, 3	8, 9, 10, 7	7, 9, 8, 6
DE	F2	8, 4, 3, 5	8, 6, 5, 10	8, 10, 6, 5
GA	F2	8, 4, 3, 5	8, 5, 6, 2	8, 9, 6, 7
PSO	F2	4, 8, 3, 5	8, 10, 9, 5	8, 7, 9, 10
CMAES	F3	3, 4, 8, 2	1, 5, 2, 9	7, 6, 9, 1
DE	F3	8, 3, 4, 2	4, 8, 3, 5	8, 3, 4, 5
GA	F3	3, 8, 4, 2	8, 4, 3, 2	8, 3, 4, 2
PSO	F3	3, 4, 8, 5	8, 3, 4, 5	8, 4, 9, 5
CMAES	F4	3, 8, 4, 5	5, 3, 1, 9	6, 10, 3, 5
DE	F4	3, 8, 4, 2	3, 5, 8, 1	3, 5, 10, 2
GA	F4	8, 4, 2, 3	3, 8, 4, 5	3, 10, 4, 6
PSO	F4	3, 8, 4, 2	3, 5, 8, 4	3, 10, 5, 4
CMAES	F5	10, 8, 6, 9	9, 1, 10, 3	5, 4, 3, 7
DE	F5	9, 8, 10, 7	10, 9, 8, 7	10, 9, 8, 5
GA	F5	7, 8, 5, 6	9, 10, 1, 8	8, 6, 7, 2
PSO	F5	8, 6, 10, 7	10, 9, 2, 7	9, 10, 7, 8
CMAES	F6	4, 8, 5, 2	3, 5, 2, 4	3, 1, 2, 6
DE	F6	4, 8, 3, 2	3, 2, 5, 6	3, 2, 6, 5
GA	F6	4, 8, 10, 2	3, 2, 5, 6	3, 2, 5, 1
PSO	F6	4, 8, 5, 3	3, 2, 5, 4	3, 1, 7, 5

Table B.2 First 25% F1-F6 Top 4 Cont. Vars

Alg	Prob	Top 4 PC1	Top 4 PC2	Top 4 PC3
CMAES	F7	3, 4, 8, 2	3, 4, 8, 5	3, 4, 8, 5
DE	F7	8, 3, 4, 2	8, 4, 3, 10	3, 4, 8, 5
GA	F7	3, 4, 8, 7	3, 4, 5, 8	3, 4, 8, 5
PSO	F7	4, 3, 8, 2	3, 8, 4, 5	3, 8, 4, 9
CMAES	F8	3, 4, 5, 8	3, 4, 8, 5	2, 3, 4, 8
DE	F8	4, 3, 5, 8	4, 3, 5, 9	3, 8, 4, 2
GA	F8	3, 4, 8, 5	3, 4, 8, 5	8, 3, 2, 4
PSO	F8	3, 4, 5, 9	3, 4, 5, 8	7, 3, 4, 9
CMAES	F9	2, 9, 7, 6	4, 2, 8, 9	8, 4, 7, 2
DE	F9	6, 1, 10, 9	7, 5, 6, 3	7, 5, 6, 1
GA	F9	10, 3, 9, 6	6, 5, 1, 3	6, 5, 7, 1
PSO	F9	3, 10, 9, 1	5, 1, 3, 9	1, 5, 6, 3
CMAES	F10	1, 2, 6, 3	1, 2, 9, 3	1, 2, 9, 4
DE	F10	1, 6, 4, 2	1, 2, 4, 8	1, 2, 9, 4
GA	F10	1, 2, 9, 6	1, 2, 9, 6	1, 9, 2, 8
PSO	F10	1, 9, 2, 6	1, 9, 6, 10	1, 9, 8, 10
CMAES	F11	6, 1, 2, 3	10, 9, 7, 3	7, 10, 9, 4
DE	F11	2, 6, 1, 8	2, 6, 1, 9	6, 1, 2, 9
GA	F11	2, 3, 5, 1	3, 2, 8, 6	10, 9, 2, 8
PSO	F11	8, 6, 1, 3	9, 1, 8, 4	3, 4, 6, 9
CMAES	F12	1, 2, 6, 3	3, 9, 1, 7	8, 5, 9, 10
DE	F12	6, 1, 3, 2	2, 6, 1, 9	8, 6, 2, 1
GA	F12	6, 2, 1, 3	6, 1, 2, 3	6, 1, 3, 2
PSO	F12	1, 2, 6, 3	3, 6, 9, 2	7, 3, 9, 10

Table B.3 First 25% F7-F12 Top 4 Cont. Vars

Alg	Prob	Top 4 PC1	Top 4 PC2	Top 4 PC3
CMAES	F13	1, 6, 2, 3	5, 6, 10, 9	8, 4, 7, 5
DE	F13	6, 1, 2, 3	2, 3, 1, 4	2, 1, 4, 8
GA	F13	6, 1, 2, 3	2, 1, 6, 3	2, 4, 6, 10
PSO	F13	1, 2, 6, 3	1, 9, 2, 6	9, 2, 7, 5
CMAES	F14	1, 2, 6, 3	9, 8, 1, 2	10, 9, 2, 7
DE	F14	6, 1, 2, 3	2, 1, 8, 3	2, 1, 3, 4
GA	F14	6, 1, 2, 3	1, 2, 3, 8	2, 1, 3, 4
PSO	F14	1, 2, 6, 3	1, 2, 6, 9	9, 7, 3, 5
CMAES	F15	1, 2, 6, 3	9, 2, 6, 3	9, 7, 10, 4
DE	F15	1, 6, 2, 3	1, 6, 2, 3	1, 2, 6, 3
GA	F15	1, 2, 6, 3	2, 1, 6, 3	2, 6, 1, 3
PSO	F15	1, 2, 6, 3	6, 2, 1, 9	6, 9, 1, 2
CMAES	F16	6, 8, 7, 9	10, 7, 5, 3	5, 7, 10, 3
DE	F16	9, 3, 10, 2	10, 1, 9, 2	3, 8, 9, 10
GA	F16	4, 5, 2, 1	5, 7, 10, 4	8, 6, 9, 10
PSO	F16	9, 2, 8, 5	9, 7, 5, 2	5, 9, 10, 1
CMAES	F17	1, 2, 6, 3	7, 5, 8, 9	5, 9, 10, 8
DE	F17	6, 1, 2, 3	6, 2, 1, 3	2, 1, 3, 6
GA	F17	6, 2, 1, 3	6, 1, 2, 3	1, 2, 3, 6
PSO	F17	1, 2, 6, 3	1, 2, 3, 6	1, 2, 6, 3
CMAES	F18	1, 2, 6, 3	9, 2, 3, 4	7, 10, 9, 8
DE	F18	6, 1, 3, 2	2, 1, 6, 3	2, 6, 1, 3
GA	F18	6, 1, 2, 3	6, 2, 1, 9	2, 3, 6, 1
PSO	F18	2, 1, 6, 3	2, 6, 9, 1	9, 6, 3, 2

Table B.4 First 25% F3-F18 Top 4 Cont. Vars

Alg	Prob	Top 4 PC1	Top 4 PC2	Top 4 PC3
CMAES	F19	8, 6, 9, 10	6, 4, 2, 5	6, 10, 7, 1
DE	F19	9, 4, 1, 8	7, 5, 1, 6	5, 7, 2, 8
GA	F19	9, 4, 8, 2	7, 2, 4, 8	6, 7, 4, 5
PSO	F19	9, 2, 10, 3	6, 9, 3, 2	4, 8, 2, 3
CMAES	F20	10, 6, 8, 7	10, 8, 6, 7	10, 9, 8, 7
DE	F20	6, 7, 9, 8	7, 8, 9, 10	10, 8, 9, 6
GA	F20	7, 3, 2, 4	7, 3, 4, 6	7, 4, 3, 5
PSO	F20	10, 8, 6, 1	8, 10, 6, 1	8, 6, 10, 4
CMAES	F21	2, 5, 8, 4	7, 1, 9, 3	6, 9, 7, 3
DE	F21	2, 4, 8, 5	4, 2, 5, 8	4, 5, 2, 8
GA	F21	2, 5, 8, 4	8, 5, 2, 4	2, 4, 10, 5
PSO	F21	8, 2, 4, 5	8, 5, 4, 3	5, 8, 9, 6
CMAES	F22	3, 5, 4, 8	10, 7, 9, 5	1, 8, 4, 9
DE	F22	3, 1, 5, 2	5, 1, 6, 7	7, 5, 9, 1
GA	F22	3, 9, 5, 6	5, 3, 9, 6	9, 8, 5, 6
PSO	F22	3, 9, 5, 6	5, 3, 10, 6	4, 7, 10, 2
CMAES	F23	5, 1, 7, 10	1, 4, 9, 7	2, 5, 7, 4
DE	F23	6, 2, 7, 5	2, 10, 8, 6	10, 6, 1, 2
GA	F23	9, 8, 7, 10	10, 5, 9, 4	2, 5, 9, 3
PSO	F23	6, 5, 9, 3	10, 5, 4, 1	4, 7, 10, 2
CMAES	F24	4, 2, 9, 5	8, 3, 10, 1	10, 8, 6, 1
DE	F24	6, 4, 1, 7	1, 3, 2, 5	6, 8, 3, 10
GA	F24	9, 1, 4, 6	7, 3, 4, 1	9, 6, 2, 4
PSO	F24	9, 4, 5, 2	9, 2, 4, 5	9, 2, 4, 5

Table B.5 First 25% F19-F24 Top 4 Cont. Vars

B.3 Fitness Quartile Windows - All BBOB

Alg	Prob	25% Window	50% Window	75% Window	99% Window
CMAES	F1	2	5	10	28
DE	F1	4	6	11	41
GA	F1	2	3	6	23
PSO	F1	1	2	3	15
CMAES	F2	2	3	7	29
DE	F2	3	4	6	26
GA	F2	2	3	4	19
PSO	F2	2	3	5	21
CMAES	F3	4	11	27	65
DE	F3	4	9	21	209
GA	F3	3	6	12	70
PSO	F3	3	7	25	194
CMAES	F4	1	2	7	68
DE	F4	3	6	15	179
GA	F4	2	4	10	66
PSO	F4	2	4	12	164
CMAES	F5	1	3	9	25
DE	F5	4	9	18	62
GA	F5	3	5	11	37
PSO	F5	2	4	7	16
CMAES	F6	2	2	3	26
DE	F6	3	4	9	57
GA	F6	2	5	10	61
PSO	F6	1	2	4	41
CMAES	F7	2	4	9	33
DE	F7	4	6	12	67
GA	F7	2	4	8	77
PSO	F7	1	2	5	61
CMAES	F8	3	4	7	24
DE	F8	3	4	6	32

GA	F8	2	2	4	26
PSO	F8	1	2	3	24
CMAES	F9	2	3	6	82
DE	F9	4	5	9	42
GA	F9	2	3	4	23
PSO	F9	2	3	4	33
CMAES	F10	2	3	5	25
DE	F10	3	5	8	57
GA	F10	2	3	7	97
PSO	F10	2	2	5	72
CMAES	F11	5	8	13	42
DE	F11	5	9	22	130
GA	F11	6	11	32	185
PSO	F11	8	18	51	216
CMAES	F12	2	4	9	26
DE	F12	4	6	11	39
GA	F12	2	3	6	21
PSO	F12	1	2	3	13
CMAES	F13	3	10	18	48
DE	F13	5	12	24	111
GA	F13	3	7	14	76
PSO	F13	2	3	7	49
CMAES	F14	2	4	11	31
DE	F14	4	7	14	52
GA	F14	2	5	10	31
PSO	F14	1	3	4	21
CMAES	F15	4	11	27	67
DE	F15	4	7	21	185
GA	F15	2	6	14	85
PSO	F15	2	6	26	95
CMAES	F16	6	12	22	42
DE	F16	14	27	71	143
GA	F16	5	9	20	109

PSO	F16	8	15	32	127
CMAES	F17	3	8	16	54
DE	F17	6	14	32	159
GA	F17	3	6	13	89
PSO	F17	3	6	16	115
CMAES	F18	2	7	18	57
DE	F18	5	12	29	168
GA	F18	3	6	12	89
PSO	F18	2	5	13	107
CMAES	F19	10	32	82	148
DE	F19	5	11	37	163
GA	F19	3	6	25	178
PSO	F19	8	16	60	168
CMAES	F20	4	9	20	45
DE	F20	3	4	6	12
GA	F20	2	2	3	9
PSO	F20	2	6	15	74
CMAES	F21	3	5	8	18
DE	F21	7	13	28	69
GA	F21	3	5	9	32
PSO	F21	2	2	4	17
CMAES	F22	3	6	9	28
DE	F22	6	10	19	65
GA	F22	3	5	11	43
PSO	F22	2	2	5	28
CMAES	F23	13	40	79	119
DE	F23	21	43	83	157
GA	F23	11	23	50	180
PSO	F23	21	39	88	186
CMAES	F24	6	20	65	94
DE	F24	6	13	38	178
GA	F24	3	8	24	211
PSO	F24	4	14	72	239

Table B.6 Full BBOB Algorithm Fitness Window Generations

Appendix C

Staff Rostering Problem Additional Tables and Figures

C.1 Full Variable Ranking Tables

Rank	1PC-MSC	5PC-MSC	10PC-MSC	50PC-MSC	AllPC-MSC	ETA_1PC
141	VAR_121	VAR_1	VAR_96	VAR_1	VAR_70	VAR_121
140	VAR_65	VAR_121	VAR_121	VAR_65	VAR_1	VAR_65
139	VAR_1	VAR_65	VAR_1	VAR_135	VAR_67	VAR_51
138	VAR_96	VAR_96	VAR_65	VAR_96	VAR_91	VAR_1
137	VAR_135	VAR_135	VAR_135	VAR_91	VAR_135	VAR_33
136	VAR_60	VAR_72	VAR_119	VAR_67	VAR_65	VAR_60
135	VAR_72	VAR_119	VAR_128	VAR_72	VAR_96	VAR_129
134	VAR_128	VAR_128	VAR_67	VAR_70	VAR_68	VAR_126
133	VAR_48	VAR_48	VAR_91	VAR_48	VAR_76	VAR_110
132	VAR_119	VAR_73	VAR_72	VAR_119	VAR_60	VAR_66
131	VAR_73	VAR_91	VAR_68	VAR_128	VAR_72	VAR_96
130	VAR_51	VAR_68	VAR_48	VAR_46	VAR_121	VAR_135
129	VAR_129	VAR_60	VAR_73	VAR_121	VAR_48	VAR_22
128	VAR_91	VAR_67	VAR_60	VAR_60	VAR_133	VAR_131
127	VAR_76	VAR_76	VAR_76	VAR_133	VAR_132	VAR_8
126	VAR_110	VAR_129	VAR_87	VAR_68	VAR_2	VAR_54
125	VAR_67	VAR_10	VAR_7	VAR_129	VAR_41	VAR_34
124	VAR_8	VAR_87	VAR_55	VAR_2	VAR_87	VAR_41

123	VAR_27	VAR_51	VAR_129	VAR_41	VAR_128	VAR_105
122	VAR_68	VAR_110	VAR_10	VAR_45	VAR_45	VAR_94
121	VAR_87	VAR_45	VAR_45	VAR_120	VAR_129	VAR_72
120	VAR_22	VAR_7	VAR_46	VAR_73	VAR_119	VAR_90
119	VAR_33	VAR_55	VAR_22	VAR_55	VAR_18	VAR_27
118	VAR_46	VAR_22	VAR_110	VAR_134	VAR_13	VAR_61
117	VAR_10	VAR_27	VAR_41	VAR_87	VAR_7	VAR_91
116	VAR_7	VAR_8	VAR_51	VAR_22	VAR_141	VAR_44
115	VAR_141	VAR_46	VAR_2	VAR_18	VAR_73	VAR_141
114	VAR_55	VAR_47	VAR_47	VAR_76	VAR_29	VAR_128
113	VAR_47	VAR_41	VAR_27	VAR_10	VAR_46	VAR_10
112	VAR_45	VAR_70	VAR_141	VAR_7	VAR_55	VAR_29
111	VAR_94	VAR_141	VAR_70	VAR_106	VAR_10	VAR_7
110	VAR_75	VAR_66	VAR_106	VAR_13	VAR_22	VAR_40
109	VAR_66	VAR_106	VAR_8	VAR_29	VAR_120	VAR_55
108	VAR_106	VAR_2	VAR_75	VAR_64	VAR_115	VAR_73
107	VAR_41	VAR_75	VAR_133	VAR_141	VAR_134	VAR_134
106	VAR_131	VAR_134	VAR_29	VAR_110	VAR_86	VAR_82
105	VAR_70	VAR_33	VAR_107	VAR_27	VAR_95	VAR_6
104	VAR_134	VAR_133	VAR_13	VAR_47	VAR_98	VAR_47
103	VAR_29	VAR_29	VAR_66	VAR_8	VAR_50	VAR_17
102	VAR_107	VAR_107	VAR_134	VAR_58	VAR_66	VAR_68
101	VAR_2	VAR_94	VAR_127	VAR_117	VAR_106	VAR_124
100	VAR_126	VAR_13	VAR_94	VAR_66	VAR_58	VAR_127
99	VAR_133	VAR_131	VAR_18	VAR_115	VAR_110	VAR_87
98	VAR_127	VAR_18	VAR_126	VAR_75	VAR_30	VAR_28
97	VAR_6	VAR_127	VAR_136	VAR_57	VAR_38	VAR_111
96	VAR_136	VAR_126	VAR_131	VAR_113	VAR_117	VAR_39
95	VAR_18	VAR_136	VAR_33	VAR_50	VAR_113	VAR_75
94	VAR_108	VAR_108	VAR_108	VAR_132	VAR_64	VAR_74
93	VAR_13	VAR_113	VAR_57	VAR_89	VAR_75	VAR_119
92	VAR_61	VAR_64	VAR_120	VAR_30	VAR_47	VAR_26
91	VAR_57	VAR_95	VAR_58	VAR_94	VAR_89	VAR_138
90	VAR_113	VAR_120	VAR_64	VAR_126	VAR_85	VAR_3
89	VAR_95	VAR_57	VAR_89	VAR_95	VAR_57	VAR_125
88	VAR_50	VAR_132	VAR_113	VAR_86	VAR_108	VAR_130

87	VAR_120	VAR_115	VAR_86	VAR_107	VAR_126	VAR_99
86	VAR_44	VAR_58	VAR_50	VAR_98	VAR_32	VAR_107
85	VAR_115	VAR_50	VAR_117	VAR_136	VAR_112	VAR_45
84	VAR_64	VAR_61	VAR_115	VAR_51	VAR_23	VAR_52
83	VAR_34	VAR_6	VAR_95	VAR_85	VAR_27	VAR_31
82	VAR_117	VAR_44	VAR_44	VAR_38	VAR_8	VAR_140
81	VAR_89	VAR_86	VAR_132	VAR_42	VAR_51	VAR_48
80	VAR_98	VAR_89	VAR_98	VAR_44	VAR_44	VAR_106
79	VAR_132	VAR_125	VAR_6	VAR_127	VAR_118	VAR_36
78	VAR_81	VAR_98	VAR_38	VAR_34	VAR_94	VAR_19
77	VAR_125	VAR_117	VAR_34	VAR_108	VAR_122	VAR_108
76	VAR_20	VAR_32	VAR_32	VAR_112	VAR_131	VAR_133
75	VAR_86	VAR_38	VAR_42	VAR_32	VAR_62	VAR_46
74	VAR_58	VAR_34	VAR_125	VAR_130	VAR_33	VAR_30
73	VAR_17	VAR_81	VAR_30	VAR_23	VAR_42	VAR_23
72	VAR_42	VAR_42	VAR_61	VAR_62	VAR_139	VAR_117
71	VAR_138	VAR_85	VAR_130	VAR_6	VAR_107	VAR_67
70	VAR_130	VAR_30	VAR_85	VAR_131	VAR_130	VAR_2
69	VAR_85	VAR_26	VAR_20	VAR_118	VAR_127	VAR_76
68	VAR_26	VAR_20	VAR_138	VAR_139	VAR_88	VAR_81
67	VAR_32	VAR_130	VAR_81	VAR_33	VAR_125	VAR_122
66	VAR_39	VAR_109	VAR_62	VAR_122	VAR_136	VAR_50
65	VAR_74	VAR_138	VAR_26	VAR_20	VAR_34	VAR_20
64	VAR_122	VAR_112	VAR_112	VAR_125	VAR_6	VAR_92
63	VAR_30	VAR_122	VAR_23	VAR_116	VAR_11	VAR_100
62	VAR_23	VAR_23	VAR_118	VAR_109	VAR_49	VAR_16
61	VAR_109	VAR_17	VAR_122	VAR_104	VAR_20	VAR_4
60	VAR_38	VAR_139	VAR_109	VAR_49	VAR_138	VAR_49
59	VAR_112	VAR_88	VAR_88	VAR_61	VAR_109	VAR_95
58	VAR_139	VAR_116	VAR_49	VAR_138	VAR_116	VAR_86
57	VAR_88	VAR_118	VAR_17	VAR_26	VAR_26	VAR_71
56	VAR_116	VAR_16	VAR_16	VAR_16	VAR_104	VAR_24
55	VAR_19	VAR_62	VAR_139	VAR_88	VAR_61	VAR_136
54	VAR_140	VAR_74	VAR_39	VAR_11	VAR_16	VAR_115
53	VAR_31	VAR_39	VAR_116	VAR_81	VAR_83	VAR_63
52	VAR_28	VAR_49	VAR_19	VAR_83	VAR_59	VAR_89

51	VAR_16	VAR_19	VAR_74	VAR_14	VAR_78	VAR_15
50	VAR_40	VAR_140	VAR_140	VAR_59	VAR_103	VAR_14
49	VAR_90	VAR_31	VAR_83	VAR_17	VAR_17	VAR_59
48	VAR_118	VAR_28	VAR_104	VAR_53	VAR_102	VAR_70
47	VAR_62	VAR_53	VAR_14	VAR_12	VAR_14	VAR_139
46	VAR_124	VAR_114	VAR_31	VAR_19	VAR_12	VAR_56
45	VAR_92	VAR_83	VAR_11	VAR_39	VAR_114	VAR_102
44	VAR_54	VAR_92	VAR_114	VAR_74	VAR_81	VAR_112
43	VAR_82	VAR_104	VAR_12	VAR_25	VAR_53	VAR_42
42	VAR_114	VAR_12	VAR_53	VAR_78	VAR_39	VAR_93
41	VAR_49	VAR_40	VAR_59	VAR_103	VAR_140	VAR_113
40	VAR_99	VAR_90	VAR_92	VAR_140	VAR_99	VAR_58
39	VAR_53	VAR_82	VAR_90	VAR_102	VAR_80	VAR_98
38	VAR_100	VAR_14	VAR_82	VAR_80	VAR_52	VAR_120
37	VAR_59	VAR_54	VAR_24	VAR_114	VAR_19	VAR_5
36	VAR_83	VAR_59	VAR_28	VAR_24	VAR_25	VAR_114
35	VAR_104	VAR_99	VAR_25	VAR_92	VAR_54	VAR_64
34	VAR_52	VAR_124	VAR_40	VAR_99	VAR_74	VAR_77
33	VAR_24	VAR_11	VAR_54	VAR_90	VAR_100	VAR_53
32	VAR_5	VAR_5	VAR_99	VAR_31	VAR_90	VAR_18
31	VAR_12	VAR_24	VAR_102	VAR_52	VAR_124	VAR_57
30	VAR_14	VAR_102	VAR_78	VAR_124	VAR_24	VAR_32
29	VAR_63	VAR_100	VAR_52	VAR_63	VAR_40	VAR_13
28	VAR_102	VAR_52	VAR_124	VAR_40	VAR_63	VAR_35
27	VAR_11	VAR_78	VAR_5	VAR_123	VAR_123	VAR_43
26	VAR_71	VAR_63	VAR_103	VAR_100	VAR_92	VAR_123
25	VAR_25	VAR_103	VAR_100	VAR_54	VAR_31	VAR_78
24	VAR_78	VAR_25	VAR_63	VAR_28	VAR_37	VAR_116
23	VAR_103	VAR_71	VAR_101	VAR_101	VAR_28	VAR_80
22	VAR_111	VAR_80	VAR_71	VAR_82	VAR_35	VAR_132
21	VAR_3	VAR_101	VAR_80	VAR_43	VAR_101	VAR_9
20	VAR_80	VAR_3	VAR_3	VAR_5	VAR_3	VAR_62
19	VAR_101	VAR_111	VAR_111	VAR_35	VAR_5	VAR_104
18	VAR_123	VAR_123	VAR_123	VAR_3	VAR_82	VAR_11
17	VAR_56	VAR_84	VAR_43	VAR_111	VAR_43	VAR_85
16	VAR_105	VAR_43	VAR_84	VAR_71	VAR_84	VAR_109

15	VAR_43	VAR_37	VAR_35	VAR_37	VAR_56	VAR_103
14	VAR_35	VAR_56	VAR_37	VAR_84	VAR_111	VAR_12
13	VAR_37	VAR_36	VAR_137	VAR_137	VAR_137	VAR_101
12	VAR_36	VAR_35	VAR_56	VAR_56	VAR_71	VAR_79
11	VAR_84	VAR_105	VAR_36	VAR_36	VAR_79	VAR_37
10	VAR_137	VAR_137	VAR_105	VAR_105	VAR_36	VAR_69
9	VAR_93	VAR_69	VAR_69	VAR_79	VAR_69	VAR_84
8	VAR_79	VAR_93	VAR_93	VAR_69	VAR_105	VAR_83
7	VAR_69	VAR_79	VAR_79	VAR_77	VAR_9	VAR_25
6	VAR_9	VAR_77	VAR_77	VAR_93	VAR_93	VAR_38
5	VAR_77	VAR_9	VAR_9	VAR_9	VAR_77	VAR_137
4	VAR_4	VAR_4	VAR_15	VAR_21	VAR_21	VAR_88
3	VAR_21	VAR_21	VAR_4	VAR_15	VAR_15	VAR_21
2	VAR_15	VAR_15	VAR_21	VAR_4	VAR_4	VAR_118
1	VAR_97	VAR_97	VAR_97	VAR_97	VAR_97	VAR_97

Table C.1 Variable Rankings by PC and Partial-ETA Results

C.2 ANOVA Variable Partial-ETA and Adjusted p-value Results

Var	Rank	eta_squared	P	adj-p	BH_critical_value
VAR_121	141	0.499	3.094E-04	3.09E-06	0.011
VAR_65	140	0.474	6.194E-07	6.19E-09	0.010
VAR_51	139	0.426	5.997E-04	6.00E-06	0.014
VAR_1	138	0.367	2.073E-04	2.07E-06	0.010
VAR_33	137	0.341	1.770E-04	1.77E-06	0.016
VAR_60	136	0.340	7.067E-04	7.07E-06	0.015
VAR_129	135	0.315	8.372E-04	8.37E-06	0.016
VAR_126	134	0.307	3.650E-04	3.65E-06	0.014
VAR_110	133	0.296	3.484E-04	3.48E-06	0.014
VAR_66	132	0.295	7.206E-04	7.21E-06	0.013
VAR_96	131	0.289	5.255E-05	5.26E-07	0.014
VAR_135	130	0.284	6.307E-04	6.31E-06	0.012
VAR_22	129	0.279	4.909E-04	4.91E-06	0.013
VAR_131	128	0.274	8.312E-04	8.31E-06	0.018
VAR_8	127	0.272	3.592E-04	3.59E-06	0.016
VAR_54	126	0.257	1.004E-03	1.00E-05	0.017
VAR_34	125	0.257	9.248E-04	9.25E-06	0.015
VAR_41	124	0.256	5.261E-04	5.26E-06	0.016
VAR_105	123	0.247	1.056E-03	1.06E-05	0.018
VAR_94	122	0.245	1.222E-03	1.22E-05	0.018
VAR_72	121	0.241	4.096E-04	4.10E-06	0.015
VAR_90	120	0.237	5.408E-04	5.41E-06	0.017
VAR_27	119	0.235	4.833E-04	4.83E-06	0.017
VAR_61	118	0.231	9.051E-04	9.05E-06	0.019
VAR_91	117	0.229	7.439E-04	7.44E-06	0.019
VAR_44	116	0.227	2.802E-04	2.80E-06	0.017
VAR_141	115	0.225	1.189E-03	1.19E-05	0.017
VAR_128	114	0.223	6.511E-04	6.51E-06	0.017
VAR_10	113	0.214	3.419E-04	3.42E-06	0.018
VAR_29	112	0.213	5.940E-04	5.94E-06	0.015
VAR_7	111	0.213	1.500E-03	1.50E-05	0.019

VAR_40	110	0.212	4.025E-04	4.02E-06	0.016
VAR_55	109	0.211	3.925E-04	3.93E-06	0.017
VAR_73	108	0.208	1.143E-04	1.14E-06	0.018
VAR_134	107	0.206	9.570E-04	9.57E-06	0.020
VAR_82	106	0.205	4.770E-04	4.77E-06	0.020
VAR_6	105	0.204	9.193E-05	9.19E-07	0.017
VAR_47	104	0.203	1.618E-03	1.62E-05	0.019
VAR_17	103	0.201	6.027E-04	6.03E-06	0.019
VAR_68	102	0.199	4.724E-04	4.72E-06	0.017
VAR_124	101	0.198	1.937E-03	1.94E-05	0.018
VAR_127	100	0.197	8.891E-04	8.89E-06	0.019
VAR_87	99	0.193	5.919E-04	5.92E-06	0.018
VAR_28	98	0.192	8.150E-04	8.15E-06	0.019
VAR_111	97	0.190	3.709E-04	3.71E-06	0.019
VAR_39	96	0.188	7.129E-04	7.13E-06	0.018
VAR_75	95	0.183	1.069E-03	1.07E-05	0.019
VAR_74	94	0.180	5.334E-04	5.33E-06	0.020
VAR_119	93	0.179	3.584E-04	3.58E-06	0.019
VAR_26	92	0.176	1.191E-03	1.19E-05	0.019
VAR_138	91	0.175	8.135E-04	8.14E-06	0.021
VAR_3	90	0.174	5.966E-04	5.97E-06	0.020
VAR_125	89	0.174	6.694E-04	6.69E-06	0.020
VAR_130	88	0.173	1.152E-03	1.15E-05	0.020
VAR_99	87	0.173	1.739E-03	1.74E-05	0.021
VAR_107	86	0.170	1.287E-03	1.29E-05	0.020
VAR_45	85	0.168	7.217E-04	7.22E-06	0.018
VAR_52	84	0.168	9.335E-04	9.34E-06	0.021
VAR_31	83	0.166	1.371E-03	1.37E-05	0.020
VAR_140	82	0.165	1.834E-03	1.83E-05	0.021
VAR_48	81	0.165	5.543E-04	5.54E-06	0.019
VAR_106	80	0.164	4.114E-04	4.11E-06	0.018
VAR_36	79	0.162	8.087E-04	8.09E-06	0.024
VAR_19	78	0.161	4.870E-04	4.87E-06	0.022
VAR_108	77	0.160	4.436E-04	4.44E-06	0.021
VAR_133	76	0.156	3.532E-04	3.53E-06	0.017
VAR_46	75	0.155	1.163E-03	1.16E-05	0.020

VAR_30	74	0.153	2.826E-04	2.83E-06	0.018
VAR_23	73	0.152	1.355E-03	1.35E-05	0.021
VAR_117	72	0.150	7.052E-04	7.05E-06	0.019
VAR_67	71	0.149	1.051E-03	1.05E-05	0.021
VAR_2	70	0.147	4.659E-04	4.66E-06	0.016
VAR_76	69	0.145	7.565E-04	7.56E-06	0.020
VAR_81	68	0.142	1.119E-03	1.12E-05	0.021
VAR_122	67	0.141	1.213E-03	1.21E-05	0.023
VAR_50	66	0.139	1.009E-03	1.01E-05	0.021
VAR_20	65	0.138	8.034E-04	8.03E-06	0.021
VAR_92	64	0.136	1.348E-03	1.35E-05	0.023
VAR_100	63	0.136	1.166E-03	1.17E-05	0.023
VAR_16	62	0.131	7.593E-04	7.59E-06	0.020
VAR_4	61	0.131	2.801E-03	2.80E-05	0.025
VAR_49	60	0.128	1.548E-03	1.55E-05	0.021
VAR_95	59	0.121	5.936E-04	5.94E-06	0.021
VAR_86	58	0.119	6.988E-04	6.99E-06	0.019
VAR_71	57	0.118	2.096E-04	2.10E-06	0.023
VAR_24	56	0.116	5.188E-04	5.19E-06	0.023
VAR_136	55	0.111	1.017E-03	1.02E-05	0.024
VAR_115	54	0.108	1.923E-03	1.92E-05	0.022
VAR_63	53	0.107	1.424E-03	1.42E-05	0.024
VAR_89	52	0.104	1.759E-03	1.76E-05	0.023
VAR_15	51	0.104	6.549E-03	6.55E-05	0.031
VAR_14	50	0.103	4.751E-04	4.75E-06	0.020
VAR_59	49	0.103	2.056E-03	2.06E-05	0.026
VAR_70	48	0.102	7.648E-04	7.65E-06	0.021
VAR_139	47	0.099	5.932E-04	5.93E-06	0.022
VAR_56	46	0.097	1.071E-03	1.07E-05	0.024
VAR_102	45	0.096	1.759E-03	1.76E-05	0.025
VAR_112	44	0.096	2.453E-03	2.45E-05	0.023
VAR_42	43	0.095	8.438E-04	8.44E-06	0.024
VAR_93	42	0.094	1.753E-04	1.75E-06	0.027
VAR_113	41	0.094	3.980E-04	3.98E-06	0.023
VAR_58	40	0.093	9.054E-04	9.05E-06	0.022
VAR_98	39	0.090	6.999E-04	7.00E-06	0.022

VAR_120	38	0.088	4.588E-04	4.59E-06	0.024
VAR_5	37	0.086	1.776E-03	1.78E-05	0.024
VAR_114	36	0.085	6.890E-04	6.89E-06	0.024
VAR_64	35	0.084	1.392E-03	1.39E-05	0.025
VAR_77	34	0.081	6.424E-03	6.42E-05	0.029
VAR_53	33	0.081	1.344E-03	1.34E-05	0.020
VAR_18	32	0.081	1.065E-03	1.07E-05	0.023
VAR_57	31	0.080	1.978E-03	1.98E-05	0.022
VAR_32	30	0.080	1.444E-03	1.44E-05	0.025
VAR_13	29	0.076	9.292E-04	9.29E-06	0.023
VAR_35	28	0.071	2.296E-03	2.30E-05	0.025
VAR_43	27	0.070	2.057E-03	2.06E-05	0.027
VAR_123	26	0.070	2.506E-04	2.51E-06	0.024
VAR_78	25	0.069	2.704E-03	2.70E-05	0.028
VAR_116	24	0.069	3.983E-03	3.98E-05	0.028
VAR_80	23	0.066	6.378E-04	6.38E-06	0.024
VAR_132	22	0.065	1.733E-03	1.73E-05	0.024
VAR_9	21	0.063	1.623E-03	1.62E-05	0.029
VAR_62	20	0.059	2.315E-03	2.32E-05	0.026
VAR_104	19	0.058	2.451E-03	2.45E-05	0.026
VAR_11	18	0.055	2.278E-03	2.28E-05	0.026
VAR_85	17	0.054	1.560E-03	1.56E-05	0.025
VAR_109	16	0.049	1.994E-03	1.99E-05	0.025
VAR_103	15	0.047	1.972E-03	1.97E-05	0.028
VAR_12	14	0.047	2.873E-03	2.87E-05	0.028
VAR_101	13	0.045	1.004E-03	1.00E-05	0.026
VAR_79	12	0.045	4.225E-03	4.23E-05	0.026
VAR_37	11	0.043	2.058E-03	2.06E-05	0.026
VAR_69	10	0.039	1.817E-03	1.82E-05	0.030
VAR_84	9	0.032	3.379E-03	3.38E-05	0.029
VAR_83	8	0.032	2.209E-03	2.21E-05	0.028
VAR_25	7	0.030	2.567E-03	2.57E-05	0.029
VAR_38	6	0.030	1.320E-03	1.32E-05	0.028
VAR_137	5	0.026	4.411E-03	4.41E-05	0.030
VAR_88	4	0.022	1.444E-03	1.44E-05	0.028
VAR_21	3	0.016	2.400E-03	2.40E-05	0.029

VAR_118	2	0.013	3.232E-03	3.23E-05	0.030
VAR_97	1	0.000	0.000E+00	0.00E+00	0.000

Table C.2 ANOVA Variable Partial-ETA and Adjusted p-value Results

C.3 Variable Most Frequent Final Roster Over 100 Runs

Variable	Roster	Variable	Roster	Variable	Roster	Variable	Roster
1	17	36	83	71	96	106	4
2	22	37	70	72	63	107	35
3	28	38	19	73	99	108	15
4	50	39	7	74	60	109	25
5	79	40	52	75	37	110	3
6	15	41	87	76	90	111	25
7	32	42	51	77	22	112	37
8	73	43	92	78	47	113	17
9	51	44	75	79	3	114	50
10	63	45	19	80	95	115	39
11	6	46	19	81	45	116	66
12	91	47	67	82	33	117	69
13	18	48	65	83	69	118	63
14	2	49	7	84	31	119	41
15	24	50	58	85	15	120	63
16	79	51	86	86	81	121	78
17	59	52	57	87	70	122	67
18	5	53	55	88	66	123	30
19	90	54	47	89	85	124	18
20	75	55	24	90	18	125	28
21	59	56	39	91	48	126	40
22	72	57	99	92	66	127	17
23	43	58	97	93	72	128	68
24	70	59	53	94	84	129	47
25	10	60	39	95	14	130	19
26	58	61	26	96	23	131	46
27	51	62	90	97	0	132	12
28	84	63	13	98	7	133	45
29	97	64	23	99	94	134	41
30	4	65	52	100	88	135	91
31	25	66	52	101	52	136	22
32	12	67	6	102	9	137	93

33	71	68	13	103	87	138	59
34	61	69	84	104	32	139	88
35	69	70	30	105	8	140	10
						141	76

Table C.3 Variable Most Frequent Final Roster Over 100 Runs