2023

Understanding Optimisation Processes with Biologically-Inspired Visualisations

Walter, Mathew J.

UNIVERSITY OF
PLYMOUTH

Understanding Optimisation Processes with Biologically-Inspired Visualisations

by

Mathew J. Walter

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

DOCTOR OF PHILOSOPHY

School of Engineering, Computing and Mathematics

December 2022

Acknowledgements

I am thankful to my family and friends for their support over the many years. I especially acknowledge the support from my parents.

I am grateful to Professor Gabriela Ochoa and Dr Sean Walton for agreeing to examine this thesis.

Finally, I want to express my profound gratitude and make a special thanks to my supervisors, Dr David Walker and Dr Matthew Craven, for their wisdom and positive encouragement throughout the project.

Author's declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Doctoral College Quality Sub-Committee. Work submitted for this research degree at the University of Plymouth has not formed part of any other degree either at the University of Plymouth or at another establishment.

Word count for the main body of this thesis: **53629**

Signed: **Mathew Walter**

Date: **December 16, 2022**

Publications and conference presentations:

Walter M. J., Walker, D. J., Craven M. J., *Visualising Evolution History in Multi-and Many-Objective Optimisation* Parallel Problem Solving with Computing conference 2020, 299-312. DOI: https://doi.org/10.1007/978-3-030-58115-2_21

Walter M. J., Walker, D. J., Craven M. J., *Visualising Population Dynamics to Examine Algorithm Performance* IEEE Transactions on Evolutionary Computing journal 2022, 1501-1510. DOI: https://doi.org/10.1109/TEVC.2022.3157143

Walter M. J., Walker, D. J., Craven M. J., *An Explainable Visualisation of the Evolutionary Search Process* The Genetic and Evolutionary Computation Conference 2022, 1794-1802. DOI: https://doi.org/10.1145/3520304.3533984

Walter M. J., Manikowski P., Walker, D. J., Craven M. J., *Explainable Optimisation of Offshore Wind Farms* A talk at the Partnership for Research in Marine Renewable Energy 2022

Walker, D. J., Craven M. J., Walter M. J., Manikowski P., *Explaining Optimisation: Applying Explainable AI Principles to Metaheuristics* Handbook of Formal Optimization Methods 2023

Abstract

Understanding Optimisation Processes with Biologically- Inspired Visualisations
Mathew J. Walter

EVOLUTIONARY algorithms (EAs) constitute a branch of artificial intelligence utilised to evolve solutions to solve optimisation problems abound in industry and research. EAs often generate many solutions and visualisation has been a primary strategy to display EA solutions, given that visualisation is a multi-domain well-evaluated medium to comprehend extensive data. The endeavour of visualising solutions is inherent with challenges resulting from high dimensional phenomenons and the large number of solutions to display. Recently, scholars have produced methods to mitigate some of these known issues when illustrating solutions. However, one key consideration is that displaying the final subset of solutions exclusively (rather than the whole population) discards most of the informativeness of the search, creating inadequate insight into the black-box EA.

There is an unequivocal knowledge gap and requirement for methods which can visualise the whole population of solutions from an optimiser and subjugate the high-dimensional problems and scaling issues to create interpretability of the EA search process. Furthermore, a requirement for explainability in evolutionary computing has been demanded by the evolutionary computing community, which could take the form of visualisations, to support EA comprehension much like the support explainable artificial intelligence has brought to artificial intelligence.

In this thesis, we report novel visualisation methods that can be used to visualise large and high-dimensional optimiser populations with the aim of creating greater interpretability during a search. We consider the nascent intersection of visualisation and explainability in evolutionary computing. The potential high informativeness of a visualisation method from an early chapter of this work forms an effective platform to develop an explainability visualisation method, namely the population dynamics plot, to attempt to inject explainability into the inner workings of the search process. We further support the visualisation of populations using machine learning to construct models which can capture the characteristics of an EA search and develop intelligent visualisations which use artificial intelligence to potentially enhance and support visualisation for a more informative search process.

The methods developed in this thesis are evaluated both quantitatively and qualitatively. We use multi-feature benchmark problems to show the method's ability to reveal specific problem characteristics such as disconnected fronts, local optima and bias, as well as potentially creating a better understanding of the problem landscape and optimiser search for evaluating and comparing algorithm performance (we show the visualisation method to be more insightful than conventional metrics like hypervolume alone). One of the most insightful methods developed in this thesis can produce a visualisation requiring less than 1% of the time and memory necessary to produce a visualisation of the same objective space solutions using existing methods. This allows for greater

scalability and the use in short compile time applications such as online visualisations. Predicated by an existing visualisation method in this thesis, we then develop and apply an explainability method to a real-world problem and evaluate it to show the method to be highly effective at explaining the search via solutions in the objective spaces, solution lineage and solution variation operators to compactly comprehend, evaluate and communicate the search of an optimiser, although we note the explainability properties are only evaluated against the author's ability and could be evaluated further in future work with a usability study. The work is then supported by the development of intelligent visualisation models that may allow one to predict solutions in optima (importantly local optima) in unseen problems by using a machine learning model. The results are effective, with some models able to predict and visualise solution optima with a balanced F_1 accuracy metric of 96%.

The results of this thesis provide a suite of visualisations which aims to provide greater informativeness of the search and scalability than previously existing literature. The work develops one of the first explainability methods aiming to create greater insight into the search space, solution lineage and reproductive operators. The work applies machine learning to potentially enhance EA understanding via visualisation. These models could also be used for a number of applications outside visualisation. Ultimately, the work provides novel methods for all EA stakeholders which aims to support understanding, evaluation and communication of EA processes with visualisation.

Contents

List of Figures

Chapter 1

Introduction

In this chapter, we describe and motivate the research problems this thesis addresses. The chapter then provides the thesis structure, delivering an overview of the content. Then the contributions of the thesis to the research literature are highlighted. Finally, the chapter is summarised.

THIS thesis aims to address multiple research problems centred around the theme of evolutionary computing visualisation. Visualisation can be an effective way to communicate large and complicated information, such as the process or solutions of an evolutionary algorithm (EA). Effective visualisation within the EA literature hardly needs motivating. Without effective visualisation techniques in the EA domain, we would make EA-centered data less accessible and far more complex, and hence slower to interpret and communicate.

In recent years a number of visualisations to visualise the approximation set (the set of final non-dominated solutions evolved by the EA) for multi-objective problems have appeared in the literature. However, by only visualising the final approximation set, we exclude most of the information about the search process. Unfortunately, scaling these visualisations to whole populations is non-trivial as the number of solutions/data points can significantly increase complexity. Furthermore, the occurrence of high-dimensional data phenomena when using multi- and many-objective solutions exacerbates the difficulty of the problem. A small amount of existing literature provides methods of visualising high-dimensional populations using dimensional reduction techniques. However, most visualisations produced by dimension reduction techniques still do not elucidate the visualisation for large or even default population sizes, especially when solutions between generations have identical or similar positions in the search space, as they can appear to overwrite each other when reduced. In most cases, the large datasets are difficult to visualise with solutions overlaying each other, causing difficulties interpreting the visualisation.

The work of De Lorenzo et al. (2019), which we later extend, provides a framework to visualise solutions of a population, preventing inter-generation overwriting, for single objective multi-parameter problems using several dimension reduction tools. The work contrasted different dimension reduction techniques. In their analysis, the most effective dimension reduction results were produced with multi-dimensional scaling (MDS) and principal component analysis (PCA). Unfortunately, the high computing complexity of MDS and PCA limits the use of this method to a small number of function evaluations (i.e., small populations). To this end, there exists an informative way of visualising the whole EA population during the search, potentially showing enormous amounts of information in a single visualisation. However, the significant computing complexity

limitations of this method and the requirement for large population sizes/runtime to converge on the problems reduce the choice of problems that could be utilised, making this method unpractical for many benchmark problems (and many real-world problems).

In Chapter 3, we extend the framework of De Lorenzo et al. (2019) to create a visualisation capable of visualising whole large populations. We develop a method with lower complexity than the original work; we are then able to visualise multi- and many-objective problem solutions rather than the simpler single-objective problem solutions used in the work of De Lorenzo et al. (2019). We evaluate this method across multiple problems and show it to be effective and informative. However, we note the method has some limitations of interpretability related to reorientation during the MDS stages. The results reveal informative mappings between the problem solutions and the reduced problem solutions in the visualisation, exhibiting information about the problem and algorithm artefacts. We then compare the visualisation informativeness against an existing indicator (the hypervolume), showing how many of these problem and algorithm artefacts cannot be detected by the hypervolume alone because of the inability to consider the search at the (local) individual solution level but can be detected in the visualisation.

We then considered the most significant limitation of the previous visualisation method, the orientation issue, and developed a visualisation that can eliminate this issue and still be used for arbitrary runtime use with a potentially high degree of accuracy by implementing an MDS approximator. We evaluate the quality of the visualisation using large populations evolved on feature-rich multi- and many-objective problems and are able to create a mapping of algorithm artefacts and problem features for use with this methodology. Then we evaluate the time complexity of the method to discover a non-linear time accuracy relationship. Finally, we test this visualisation for a number of different EAs with different parameters and evaluate how effective this visualisation can be at visually communicating and evaluating algorithms and the effects of algorithm parameters. This section of work concludes by having established and evaluated a visualisation method, which mitigates the research problems we initially highlighted to be used for arbitrary runtime use and to potentially create informative projections of the entire population in a single visualisation. We can now explore other adaptions and applications of this visualisation.

The evolutionary algorithm search is often difficult to understand, even for EA researchers. Usually, the evaluation of search performance is conducted by considering the approximation set, providing direct information only about the final solutions. However, more is needed to assist one with understanding the inner workings of an EA during the search, the problem-algorithm interaction, the direct effect of parameters, the robustness of the solutions, debugging, traceability and so on, in order to enhance the interpretability of the process and results. These challenges disaffect EA developers, practitioners and other stakeholders. Consequently, several recent works have called out a demand for more explainability in the field of EC. Furthermore, a workshop on evolutionary computing explainable artificial intelligence (ECXAI) at the Genetic and Evolutionary Computation Conference (GECCO) 2022 attracted large interest and brought on a further call for a demand for explainability within EC.

The use of explainable AI (XAI) has brought many benefits to AI users. It has assisted

with propelling the use of AI in some industries through better understanding and trust in the decision-making process, accelerating the fourth industrial revolution. Whilst XAI methods have been influential in the XAI literature, most methods cannot be trivially cut and pasted over to the ECXAI domain (because of a difference in what is to be explained). Moreover, as of 2022, very few explainability methods exist for EC. Due to a lack of ECXAI literature, we motivate the requirement for ECXAI and the many benefits it could bring to the field. In Chapter 4, we then develop one of the first methods (PopDP) of providing explainability to EC, specifically creating an explainability method for EAs.

Visualisation has already been a crucial tool for interpretability within AI providing many benefits to AI users, such as the ability to project a large amount of information in a single figure. We develop and show a potentially highly informative visualisation in Chapter 3. The visualisation appears to be an effective foundation for building an explainable tool by utilising the existing informativeness of the visualisation and then aiming to develop greater explainability without overwhelming the user. Unfortunately, not all visualisation is explainable; for a visualisation to be explainable, it must reveal the inner workings of the decision-making process understandably (e.g., a heatmap of the solutions provides a very low degree of understanding about the inner workings of the search). However, we note the threshold of understanding for a method to be classed as explainable is debatable, as there is still no single agreed definition of explainability even in the XAI literature. The explainable visualisation method we develop is defined as PopDP, and we evaluate its explainability on benchmark problems. Finally, we apply and evaluate the method on a real-world offshore wind farm problem. We conclude by arguing why this method is explainable and ways of quantifying this explainability. We also discuss the future of explainability in evolutionary computing.

Despite the contributions made in Chapter 3 and Chapter 4 toward better interpretability and explainability of the search, detecting local optima in a high-dimensional space after a dimension reduction projection is often non-trivial, with local optima detection being very important for algorithm convergence. In Chapter 5, we consider a novel development of methodology which uses machine learning to produce models which can be trained to learn EA characteristics as they evolve solutions through problems. The models can then be used on unseen/untrained solutions to predict whether a solution from the same metaheuristic is in a region of optima. Given the effects optima (particularly local optima) can have on the algorithm search landscape interaction, the EA community benefits in being able to quickly and autonomously detect local optima. Moreover, these novel optima detection models have many potential applications, such as building self-adaptive hyperparameter EAs based on real-time algorithm performance. However, in this work, we consider the models to maximise visualisation informativeness by using ML to intelligently learn model characteristics and make predictions which can be visualised by colouring the solutions in the visualisation based on the prediction of whether the solution is in a region of optima or not - creating an intelligent visualisation. We again use the developments in Chapter 3, to provide a framework to plot the high-dimensional solutions in a manageable way. The work was a proof of concept rather than an attempt to create an optimal-accuracy model. We developed the models on limited algorithms and multi- and many-parameter problems; however, we provide the results and evaluation to show that this proof of concept -

that machine learning can be used to predict algorithm characteristics - is compelling. Finally, we discuss the limitations of the work and future work projection to overcome these limitations, increasing the scalability of the work.

To summarise this thesis, we develop and evaluate two potentially informative visualisations that can be used for arbitrary runtimes and can show the population evolving through the search space with high accuracy. We use the visualisation as a basis to develop one of the first explainable evolutionary computing methods, namely PopDP. The work may be useful for EA users who wish to consider methods that provide greater interpretability and explainability of an EA search to capitalise on the many benefits greater interpretability brings. We then use machine learning to establish EA characteristics predictive models to build intelligent visualisations. The models are useful for users who wish to intelligently/autonomously predict EA/solution characteristics for potentially enhancing and understanding the EA search; alternatively, one may be able to develop other new EA technologies based on the machine learning model framework.

1.1 Thesis Structure

The thesis structure takes the following foundations:

Chapter 2: Background The background chapter introduces the relevant definitions, methods and literature required for the foundations of this thesis. The background covers the introduction to evolutionary algorithms, the existing explainable AI literature and the existing EA visualisation literature.

Chapter 3: Visualising Evolution History in Multi- and Many-Objective Optimisation This chapter begins by considering an existing population visualisation and the limitations of the method. The chapter then mitigates these issues and develops a potentially informative visualisation (two different methodologies) which can be used for arbitrary runtime use, highly accurate, informative and scalable. This chapter evaluates this visualisation qualitatively and quantitatively, utilising multi-feature benchmark problems in a range of objectives. These developments provide the basis of a visualisation for the following chapters.

Chapter 4: An Explainable Visualisation of the EA Search Process The chapter aims to provide an explainable visualisation tool for interpreting the search of evolutionary algorithms. The visualisation is applied to a real-world offshore windfarm problem and evaluated to show the explainability benefits. We also motivate and provide a discussion on the future of the nascent explainable evolutionary computing domain.

Chapter 5: Intelligent Visualisation: Enhancing Optimisation Visualisations with Deep Learning This chapter proposes the use of machine learning to enhance evolutionary algorithms. The work considers machine learning to predict search characteristics to build intelligent visualisations. The models and visualisations are both evaluated on single-, multi- and many-parameter problems.

Chapter 6: Conclusion This chapter conjoins elements from the preceding chapters to conclude the work and future work.

1.2 Thesis Contributions

The primary contributions of this thesis are the following:

1. A comprehensive review of the current EA literature linking visualisation and explainability (Chapter 2).
2. We provide two methods to extend a single objective parameter space visualisation to multi- and many-objective visualisation methods, overcoming complexity issues and allowing one to scale with arbitrary runtime use (Chapter 3).
3. We evaluate the visualisation methods developed in Chapter 3 against a benchmark problem set to conceive a comprehensive mapping of the population, problem and algorithm features in the visualisation (Chapter 3).
4. We develop one of the first methods to visualise the evolutionary search process and which aims to provide a degree of explainability (Chapter 4).
5. We develop a method for capturing the characteristics of an EA population evolving through a problem - with possible widespread application potential - we create a predictive model to indicate when a solution is in a region of optima (Chapter 5).
6. We use these predictive models to develop intelligent visualisation using the ML model to enhance visualisations. We evaluate these models to gain insight into the model workings (Chapter 5).

1.3 Summary

In this introduction, we have motivated the research problems considered by the work in this thesis at a high level. We also define how this thesis is structured and the key novel contributions made by this thesis. Next, we introduce the relevant background material.

Chapter 2

Background

The background chapter provides a basis of knowledge on which the rest of this thesis is formulated. First, the motive for implementing evolutionary algorithms is introduced, and then the common definitions, techniques and methodologies are explored. Next, the following section motivates explainable artificial intelligence considerations for this work. Finally, evolutionary algorithm visualisation techniques from the existing literature are reviewed.

THIS chapter discusses three significant areas of artificial intelligence upon which this work is predicated; thus, this chapter is split into three sections. We start by introducing the motive for evolutionary algorithms (EAs) in optimisation. We then dissect the mechanics of evolutionary algorithms to consider the algorithm methods and performance indicators. We then review common well-established multi- and many-objective evolutionary algorithms before examining test problem benchmarks.

In the second section, we motivate the importance of explainability in artificial intelligence and within the nascent field of explainable evolutionary computing to consider the key advantages to incorporate and the disadvantages to mitigate in designing methods in the later chapters. We also examine definitions and the current direct, post-hoc, model-specific and model-agnostic explainability methods. The section also considers the existing explainable evolutionary computing literature.

In the final section, we review the existing EA visualisation literature, with a focus on solution and population visualisation methods. We examine the methods to consider the advantages and disadvantages of the existing visualisations, which we later contemplate when proposing new visualisation techniques in the technical chapters. These three sections will provide a basis of knowledge for the introduction of the later technical chapters.

2.1 Optimising Multiple Criteria

When deciding on a car to purchase, multiple criteria often need to be considered and either maximised or minimised. For example, a car with a large engine capacity may be faster than a car with a smaller engine capacity but with less efficient fuel consumption. In another example, the more stylish the vehicle the less affordable it may be. As some of the criteria are conflicting, to achieve the best criterion values in all criteria, it will require one to find a car with minimal sacrifice in every criterion - this is the principle of optimisation. When comparing the criteria values of numerous cars against each other it could be that no single car yields the best values in all criteria. However, it may be the case that one could exclude a subset of the original choices containing the cars which are beaten (a better score with respect to maximisation or minimisation) by another car

in all the criteria considered. The purpose is to obtain a smaller (with the best criteria) set of cars to choose from by optimising the choice of cars based on criteria.

Without optimisation we would only be able to choose the solutions that were best on all criteria (often infeasible for many problems), on a single criterion or randomly choose the solutions. It is trivial to extend this principle of optimisation to everyday life, and it is vital in the domains of science and industry. Applications of optimisation include but are not limited to telecommunications (Alba & Chicano 2006), cyber security for heuristic-based (opcodes) malware detection (Manavi & Hamzeh 2019), engineering (electric car velocity planning) (Targosz et al. 2013), chemistry (structure prediction of molecular crystals) (Zhu et al. 2012), biology and medicine such as computational drug design (choosing protease inhibitor candidates for SARS-CoV-2 vaccines) (Bäck et al. 2020).

We now define some of the nomenclature. In the case of the car analogy, the individual cars would be represented by a set of *parameters* and lie in *decision space* or *parameter space*. We note ambiguities with the word parameter in the EA domain; parameter can also refer to EA hyper-parameters (optimiser variables). The cars have criteria values such as engine capacity (litres) or fuel efficiency (miles per gallon). These values are referred to as the *fitness* value or *objective* value and can hold different units of measurement in the different objectives (such as litres and miles per gallon). When comparing the objectives of two solutions, the units should be identical or scaled appropriately. Objectives may also be weighted based on their importance. The fitness is found in the fitness/objective space. The fitness of a solution is measured using an evaluation function called the fitness function or objective function.

Optimisation problems can be categorised into the number of criteria for the problem. A problem with only a single criterion to optimise is defined as a single-objective problem. A problem with two or three objectives is defined as a multi-objective problem (MOP) and a problem with more than three objectives is defined as a many-objective problem (MaOP). An optimisation problem can maximise or minimise objectives or use a combination for MOPs and MaOPs. The task for all these problems is to optimise a problem comprising a set of M conflicting objectives to which no solution can simultaneously optimise all M objectives.

More formally, to define the definitions above, a multi-objective optimisation problem comprises M competing objectives, such that a solution x is quantified by an objective vector y with M elements

$$\begin{aligned} y &= (f_1(x), \ldots, f_M(x)) \\ &\text{such that } x \in \Omega,\ y \in \Lambda, \end{aligned} \tag{2.1}$$

where Ω is the *search space* and Λ is the *objective space.* As stated, many-objective optimisation problem comprises four or more competing objectives (and thus $M \geq 4$ for such problems).

Solutions in which all corresponding fitness values are better than the other solutions' fitness values are more optimal. The solutions possessing better fitness values in all objectives are said to *dominate* solutions with weaker fitness values. It can also be the case that a solution holds better fitness values than another solution in some objectives but worse in other objectives. These solutions are referred to as *mutually*

non-dominating. The working set of solutions that are generated at the current generation is defined as the *population*. The solutions that are mutually non-dominating make up a subset of the population called the *archive*. The final non-dominating solutions' in the objective space are defined as the *Pareto front*, and the corresponding solutions in the decision space are referred to as the *Pareto set*. Challenges arise comparing these non-dominating solutions (Farina & Amato 2002).

Solutions are compared using the dominance relation, whereby solution y_i dominates solution y_j if it is no worse than y_j on any objective and better on at least one. More formally, assuming a minimisation problem without loss of generality:

$$y_i \prec y_j \iff \forall M(y_{iM} \leq y_{jM}) \wedge \exists M(y_{iM} < y_{jM}). \tag{2.2}$$

If neither y_i dominates y_j, nor y_j dominates y_i, then the solutions are mutually non-dominating. A solution with no dominating solutions is non-dominated. The goal of an optimiser is to identify the Pareto set, the set of solutions in the decision space (Ω) that cannot be dominated. The objective space Λ image is called the Pareto front.

A Pareto front and Pareto set can therefore be mathematically defined:

$$PF = \{\mathbf{y} \in \Lambda | \neg \exists \mathbf{y}^{'} (\mathbf{y}^{'} \in \Lambda \wedge \mathbf{y}^{'} \prec \mathbf{y})\} \tag{2.3}$$

$$PS = \{\mathbf{x} \in \Omega | f(\mathbf{x}) \in PF\} \tag{2.4}$$

2.2 Evolutionary Algorithms

Early conventional optimisation methods often involved linear programming or hill climbers (an algorithm that mutates a solution, then compares the solution's fitness value to a previous solution's fitness and selects the best solution of the two). While these elementary methods are usually fast to converge, they are limited by difficulties overcoming local optima and being inadequate at providing a diverse set of solutions. Moreover, often problems do not have gradient functions (such as for a UAV finding an optimal path online (Nikolos et al. 2003)), the problems may not have easily differentiable gradients or no exact approaches can be devised, rendering conventional optimisation methods impracticable. Further information on gradient based optimisation can be found in the work of Walton (2013).

Within the branch of Artificial Intelligence and the sub-branch of Evolutionary Computing lies a type of optimiser defined as Evolutionary Algorithms (EAs). EAs were developed to address and overcome the challenges associated with conventional optimisation methods. The principle of an EA is to generate an initial solution or set of solutions and then perturb the solution(s) through the search space using nature-inspired operators to reach a desired global optimum/optima. An optimum is an optimal point in the search space (singular), and optima are the plural of optimum. When a solution has no better neighbours, that is none of the nearest surrounding solutions has an improved objective value, the solution is in a region of optimum/optima. If no other solutions in the total objective space are better than the solution, then the solution is in a region of global optimum/optima. If neighbouring solutions yield worse objective values but solutions within the whole search space yield better objective values, then the solution is in a region of local optimum/optima.

Evolutionary algorithms are based on the learning from several million years of evolutionary biology. In a genetic algorithm (a type of EA), each solution is made up of genes, and the solution's genotypes (decision variables) encode its phenotypes (fitness) via a mapping (fitness function). A specific gene value is also known as an allele, and a set of genes required to create a phenotype is called a chromosome. Just as the evolutionary process of homo sapiens relies on genetic mutations and genetic crossover, evolutionary algorithms simulate biological crossover and mutation to evolve solutions to which selection pressure eliminates the weaker solutions from the population. Along with the selection pressure, the evolutionary operators will intend to produce solutions that converge as best as possible to the Pareto front.

In order to obtain a more diverse selection of solutions, the operators also need to enhance diversity among solutions. Diversity creates a better spread of solutions across the frontier, ensuring solutions are not all converged to the same small area on the frontier, giving the person choosing the final solution to implement from a set of solutions, known as the decision-maker (DM), more choice to choose optimal solutions in more areas of the search space. When an approximation Pareto front has been generated at the end of an EA process, a DM can decide which solution(s) they wish to use, and the solution(s) corresponding decision values should be obtainable. Note no single measure for diversity exists, and diversity can be discussed in both the decision and objective space.

Evolutionary algorithms can be optimised offline or online. In an offline configuration, the evolutionary algorithm is run to a termination state where the DM is presented a final set of solutions. In online optimisation, typically, the DM is present and can guide the solutions based on their preferences and expert knowledge of the problem. Based on the interaction of the DM, the EA can be classified into *a priori*, *a posteriori* or interactive methods (Sanchis et al. 2008). The *a priori* methods involve the DM providing information about solution preference before the algorithm is run, and the algorithm can focus on a specific area of the search space (i.e., by applying a weighting to the objective functions to convert the problem to a scalar objective problem (SOP)), to search a space of greatest interest to the DM. The *a posteriori* methods assume no immediate preference and provide the DM with solutions after the algorithm has reached a termination state, in which the DM receives a portfolio of solutions; the DM can then apply their preference. Interactive methods produce preliminary results for the DM allowing the DM to understand the problem, better guide the search without being overloaded with information and generate more confidence in the final solution (Miettinen 2012).

2.2.1 Methods and Operators

Operators and selection pressure aim to increase the mean fitness quality and diversity of the population; however, solutions in future generations are not always better than in previous generations due to the heuristic nature of the operators. Operators are usually divided into two types, recombination and mutation.

Recombination

Recombination or crossover operations choose two solutions (defined parent solutions) in the population (preferably solutions with high-quality fitness) and combine some pro-

portion of solution 1 with a proportion of solution 2 to produce an offspring (child solution). In theory, the offspring should possess the genes of good parents and ultimately have greater fitness than solutions in their ancestors' generations. However, due to the heuristic nature of EAs, this is not always true. Crossover usually requires choosing two solutions for parents and a crossover probability p_c which determines how much of the next generation is produced using crossover, i.e., a crossover rate of 0% means no crossover was used and the new offspring are produced from other methods. 100% crossover means all child solutions were produced using crossover. Common default crossover probabilities are set around 0.6-0.8 (Eiben et al. 2003).

Different methods exist to perform the crossover operation. For example, one of the most primal is the *one-point* crossover. One-point crossover involves choosing a single gene position in the chromosome and exchanging the genes up to that position with another chromosome. In a *two-point* crossover, two points are selected on the parent chromosomes, and genes are exchanged between these two points. Both methods result in two offspring. The *n-point* crossover cannot be directly applied to permutation-based problems (such as the Travelling Sales Problem) as duplication of alleles can occur and render the solution infeasible.

For crossover using the permutation representation, one can adopt the Partially Mapped Crossover (PMX) operator (Goldberg et al. 1985). For PMX, two substrings from each parent are chosen and exchanged at random, much like a two-point crossover. A mapping relationship is identified for duplicate genes. The solutions are then legalised by replacing duplicate genes with the mapping. Variations of PMX also exist, such as multi-parent partially mapped crossover (MPPMX) (Ting et al. 2010), which uses more than two parents to generate solutions.

Simulated Binary Crossover (SBX) was proposed to provide a real-coded crossover operator yielding similar search power to that of *single-point* crossover used for binary problems (Deb et al. 1995). This was done by providing a parameter and drawing offspring values from a twin peak probability distribution with a peak centred on each chromosome. The parameter (usually a fixed value), namely the distribution index η_c, controls the spread of offspring solutions. Therefore, larger values of η_c will provide a higher probability of generating near-parent solutions, whereas a small η_c will flatten the distribution peaks which create offspring from solutions further apart and hence increase diversity from the parent solutions (Deb & Beyer 2001).

Mutation

The mutation operator is applied to some number of chromosomes to mutate/alter a gene value in a stochastic way. Mutation reduces inbreeding by adding new members to the gene pool. This typically happens before the crossover operation, except in some cases, such as differential evolution (Storn & Price 1997). The mutation is generally probabilistic and can be distribution dependant, adding to the stochastic nature of EAs. A simple example of the mutation operator is the bit flip. This mutation operator can be applied to binary representations of a gene. The operator works by 'flipping' a 1 to a 0, or a 0 to a 1.

A gene can be mutated for continuous gene representation problems by adding some random number from an appropriate distribution to the gene value. The mutation rate

is a probability denoted p_m, which determines what proportion of the population has the mutation operation applied. Mutation rates are often set between $1/N$ and $1/\mu$ (Eiben et al. 2003), where N is the number of bits in the chromosome, and μ is the number of individuals in the population. One benefit of mutation is that it can be applied to a single parent solution and thus applied to a single solution population. Mutation can stop solutions from becoming trapped in local optima as an increase in mutation can result in greater search space exploration. Increasing the mutation probability to 1 would result in a random search (which would exhibit hill climber characteristics), whereas a mutation probability of 0 would result in no mutation.

A common mutation operator is a Gaussian mutation, which uses the Gaussian distribution ($z \sim N(0, \sigma)$) to sample mutation sizes (Yao et al. 1999). The exponential decay of the Gaussian distribution encourages small gene mutations whilst still allowing larger but less frequent gene mutations. Other distributions such as polynomial distributions can be sampled for an appropriate mutation size (Deb et al. 1996). Where permutation matters, more niche operators may be used to mutate the solutions without invalidating them from duplication, such as the swap and insert operator, to stop the replication of permutations where an item can only hold one position at a time (Eiben et al. 2003).

Selection

Once the population is initiated, objective functions and operators are defined, selection needs to be applied to choose the best quality solutions for the next generation. The purpose of selection is to apply pressure from the algorithm to the population to obtain the best quality solutions for parenthood. This choice is usually based on the solution's fitness values obtained from the fitness function. This process is usually deterministic, unlike parent selection for reproduction which is stochastic. EAs which choose solutions from both the parent and child population are elitist, and this concept is defined as elitism (Deb, Pratap, Agarwal & Meyarivan 2002).

For single objective solutions, one can trivially find the best solution(s) by finding the solution which yields the greatest value for objective maximisation or the lowest value for objective minimisation. For multi- and many-objective solutions the problem is non-trivial. For selection, a ranking is applied to order the solutions based on their fitness. For multi- and many-objective solutions, there are many types of ranking (Walker 2012), with the most common being the Pareto sorting (Deb, Pratap, Agarwal & Meyarivan 2002). Furthermore, diversity measures can be included in the selection process, such as crowding distance which is discussed in more detail later (see Section 2.2.5).

Tournament selection is one of the most common EA selection techniques, a number of solutions (called the tournament size), are compared, and the individual with the highest quality fitness value is chosen to make up the next generation population. A greater tournament size increases the pressure of selecting better quality solutions as there are more solutions in the selection pool to choose from. Often a tournament size of two solutions is used; this is denoted binary tournament selection. Another common selection technique is the proportional roulette wheel selection which chooses solutions to make up the next generation and increases the probability of an individual being chosen proportionately with the individuals' fitness quality.

2.2.2 Performance Indicators

Performance indicators are required to explicitly evaluate an algorithm's performance which can be used for analysis. Given that an EA's primary objective is to derive the highest quality solutions, one method of evaluating the quality of an EA would be to assess the quality of the final solutions the algorithm yields. Although it is difficult to evaluate other aspects of the algorithm's performance, such as its performance throughout the entire search and the robustness/applicability to other problems, when basing the entire algorithm evaluation exclusively on the final approximation set, we make this case in the later chapters. We note there is no acceptance of a single measure of algorithm success within the EA community; instead, many different metrics exist to determine algorithm performance (Zitzler et al. 2003). The work of Audet et al. (2020) identifies 63 different performance indicators. In initial works, numerical metrics are calculated to determine approximation front convergence to the Pareto front, the diversity of the solutions on the Pareto front and the distribution of non-dominated solutions on the front. These three objectives were laid out in the work of Zitzler et al. (2000), to be the conditions a Pareto set should satisfy. Moreover, evaluating the approximation set against the true Pareto set can be challenging as, for some problems, the true Pareto set is unknown.

Hypervolume

Attempts at understanding the convergence and diversity of solutions by considering the front of final solutions include the well-known work of hypervolume (Fleischer 2003). The hypervolume is one of the most prevalent metrics utilised by the research community (Riquelme et al. 2015). The hypervolume, also known as the *s-metric*, calculates a numeric real value to determine the convergence and diversity of solutions. Given a reference point such as the nadir point (the worst possible solution from a set of solutions), we calculate the volume of the space between the nadir point and the approximate Pareto front by summing the hypercubes in the dominated space between the nadir point and the non-dominated solutions. The larger the value, the greater the distance from the nadir point, indirectly increasing the likelihood of solution convergence to the optima and the diversity of the solutions. If the Pareto front is known, the hypervolume of the Pareto front and the hypervolume of the approximation front can be used to determine the hyper-area ratio.

More formally, the hypervolume can be formulated as (Brockhoff et al. 2008):

$$I_H(\mathbf{Y}) = \text{vol}\left(\bigcup_{\mathbf{y}_i \in \mathbf{Y}} [y_{i1}, \dot{y}_1] \times [y_{i2}, \dot{y}_2] \times \ldots \times [y_{iM}, \dot{y}_M]\right), \tag{2.5}$$

where $\dot{y}$ is a reference point. The hypervolume has also been integrated into EAs, such as the hypervolume estimation algorithm or HypeE, which utilises the hypervolume to improve the search process (Bader & Zitzler 2011). Whilst the merits of these metrics' are well known, the computation complexity of calculating the hypervolume is computationally heavy, creating limitations in the population size and the number of objectives. To overcome the computation limitations, Monte Carlo variations of hypervolume (Bader & Zitzler 2011) have been proposed, which provide an estimate of the dominated space through Monte Carlo sampling.

Generational Distance and Inverted Generational Distance

Other performance indicator methods include the Generational Distance metric (GD) (Van Veldhuizen & Lamont 1998). The GD requires prior knowledge of the Pareto front and measures the average distance from the approximate solutions to its nearest neighbour in the Pareto front. A metric of zero is the best possible situation in which the approximate solutions are equal in position to the true Pareto front. This is perhaps one of the most intuitive methods of measuring the error between the estimated Pareto front and the true Pareto front. A limitation of GD is its sensitivity to the number of points in the approximation front, i.e., if the algorithm returns a single solution approximation front which exists in the Pareto front, the GD will equal zero (Audet et al. 2020) whilst producing a non-diverse approximation set.

Other similar approaches include the Inverted Generational Distance (IGD) (Bosman & Thierens 2003). IGD also measures the average distance from the approximate solutions to its nearest neighbours in the Pareto front. GD and IGD hold similar advantages and disadvantages (Audet et al. 2020). More formally, the IGD can be formulated as

$$IGD(Y_N, Y_p) = \frac{1}{|Y_p|}\left(\sum_{y_2 \in y_p}\left(\min_{y_1 \in Y_N} \|y_1 - y_2\|\right)^p\right)^{\frac{1}{p}}. \tag{2.6}$$

When $p = 2$ as for most cases, equating to the average Euclidean distance between the two sets. GD/IGD are the most common metrics of solution evaluation, with IGD increasing in frequency of usage, whereas the GD indicator is decreasing in usage amongst the EA literature (Riquelme et al. 2015).

Attainment Functions

A further type of performance indicator which is primarily visualised is the empirical attainment function (EAF) (Fonseca et al. 2001); the EAF involves generating multiple approximation sets providing a probabilistic distribution of objective space outcomes from an EA (López-Ibáñez et al. 2010). This can allow for the comparison of algorithms (by plotting in separate figures) or the detection of algorithm properties statistically. In visualisation, the multiple sets are usually superpositioned over each other, resulting in darker shades representing areas that are covered by more of the sets of solutions, hence having a greater EAF value. Visualisation of EAF is usually in two objectives, but the work of Tušar & Filipič (2014*b*) considers the visualisation of EAF in three objectives extending the 2-D rectangle sets to 3-D cuboids.

2.2.3 Exploration and Exploitation

Exploration and exploitation are the two fundamental principles a search algorithm is required to employ. Exploration is the process of perturbing solutions to previously unexplored areas within the search space. Exploitation is the process of examining neighbouring solutions of good candidate solutions for convergence to optima/the optimum. A good exploration-exploitation ratio is essential for an effective search algorithm. However, this ratio is problem-dependent - for example, multi-modal problems require greater exploration than unimodal problems to stop the algorithm from becom-

ing stuck in local optima. Furthermore, different exploration-exploitation ratios are optimal at different stages of the evolution process; initially, exploration is required to search the space for suitable solutions, and then exploitation allows the EA to exploit for convergence.

Researchers often oversimplify and misunderstand the causation of exploration and exploitation as the causation of exploration and exploitation is non-trivial (Črepinšek et al. 2013). For example, the assumption selection drives the exploitation, and the search operators enhance exploration is a common misconception amongst EA practitioners. It is, therefore, better to consider selection and search operators to both hold exploration and exploitation properties (Črepinšek et al. 2013). Mutation can create solutions in unexplored regions of the search space, increasing exploration. However, most of the parental genetic material is preserved, so the mutation operator can also be considered an exploitation operator. The crossover operator uses entirely parental information to create offspring, thus can be considered exploitation; conversely, solutions created can explore new areas of the search space. Furthermore, other factors such as the population size can also impact the exploration and exploitation trade-off.

The question of how to measure exploration and exploitation in EAs is an open question (Beyer & Deb 2001). One proposed metric for calculating the exploration-exploitation trade-off is in the work of Črepinšek et al. (2013). The work presents four different methods of calculating a metric. The work of De Lorenzo et al. (2019) adapts this metric to colour visualisations based on this metric, something we continue to implement in the later chapters of this work. The metric can be defined as

$$\begin{aligned} scn(ind_{new}, \mathbf{P}^{'}) = \min\, & d(ind_{new}, ind) \\ & ind \in \mathbf{P'} \wedge (\mathbf{P'} \cup \mathbf{P}) \\ & ind_{new} \neq ind, \end{aligned} \tag{2.7}$$

where d is a distance function such as the Euclidean distance, ind is an individual from the existing population, and ind_{new} is a newly created offspring individual. P denotes the whole population, and P' is some subset of the population. In much of this work, we consider P' to be all individuals of the populations up to but not including the current generation. scn is the similarity of two individuals, previously defined as the minimal Euclidean distance between a solution and another member of the population. X is the median of minimum pairwise distances in a population. The median can be an alternative to using the mean when the data contains large outlier values.

$$scn(ind_{new}, \mathbf{P}^{'}) > X \longmapsto \text{exploration} \tag{2.8}$$

$$scn(ind_{new}, \mathbf{P}^{'}) \leq X \longmapsto \text{exploitation} \tag{2.9}$$

An increase in the Euclidean pairwise distances of solutions in a population suggests greater exploration and less exploitation, as the population is increasing the search space area explored.

Later work (Jerebic et al. 2021) considered another measure of exploration-exploitation. They compute $ExpBas$, which is a measure of exploration and exploitation by considering basins of attraction. A basin of attraction is an area of the search space which

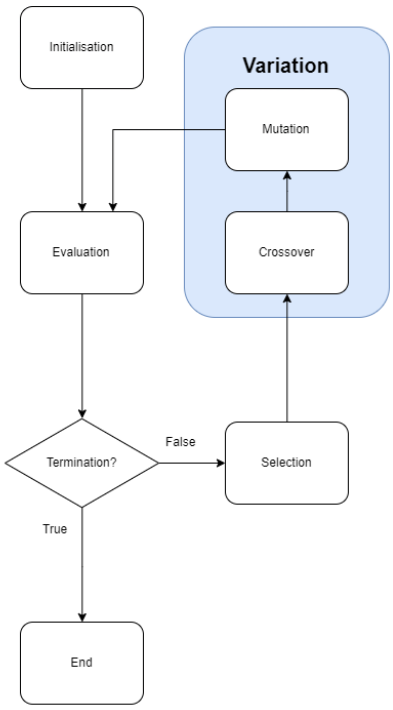


Figure 2.1: The evolutionary algorithm process chart.

would lead to an optimum when acting a local search upon. The metric is calculated by generating a heat map of objective values in the search space. Then, marking all possible boundaries of the attraction basins by checking the two closest neighbouring fitness values from opposite sides of the potential boundary point. We then identify boundaries and check which points are in similar basins of attraction. Finally, all false boundaries are removed. This allows one to develop a metric based on the principle that solutions that perturbate through the same boundary exhibit traits of exploitation.

Evolutionary Algorithm Framework

When employing an EA to optimise a problem, one first needs to decide on a suitable encoding of candidate solutions which will probably affect the choice of EA utilised. Then a series of valid random solutions should be initialised. We then evaluate each solution to determine the fitness of each solution. Based on this evaluation, we can choose parent solutions, usually solutions of best fitness and diversity. We then apply pre-defined operators (mutation and crossover), which exchange or alter solution information to generate new offspring solutions (these operators are often stochastic, not consistently generating better offspring than parent solutions). Offspring are then evaluated, and selection pressure selects solutions for the next generation. Finally, we repeat the process of evaluating solutions, choosing parents and creating offspring until a termination condition is fulfilled (such as a maximum number of evaluations or when the fitness has reached a pre-defined threshold). This process is illustrated in Figure 2.1.

Different EC methods often follow a similar structure, but the encoding of candidate solutions differs. For example, evolutionary strategy (ES) (Rechenberg 1973) solutions are composed of real-valued variables, genetic algorithms (GAs) (proposed by Holland (1975)) are composed of binary variables, and genetic programming (GP) (Koza et al. 1989) solutions are represented as graph or tree structures. So the decision of encoding the candidate solutions and hence the choice of EC method is often problem-dependent. Evolutionary algorithm performance is also problem-dependent, and thus the no free lunch theorem was suggested, which precludes the existence of a single dominating algorithm for all problems (Gao et al. 2019).

2.2.4 Genetic Algorithms

For optimising the knapsack or one max problem, a suitable encoding of solutions to the problem is an array of binary variables. Thus, a genetic algorithm would be an effective EC method to optimise these types of problems. The genotype is encoded in a binary representation for these problems, whilst the phenotype is not necessarily binary. In the case of the Knapsack problem, the genotype is a binary vector $X = (x_1, \ldots, x_n) \in \{0,1\}^n$ of length n. Each index represents a unit/item, with 1 representing a unit in the knapsack and 0 representing the absence of a unit from the knapsack. The mutation is typically applied as a bit flip operator and one-point crossover as the crossover operator (choosing a random index and combining the genetic information of two parents). There are 2^n possible combinations, so as the length n (possible units) increases, the search space size increases exponentially. Each binary solution vector is evaluated by checking the index of the binary array for values of 1, finding their corresponding fitness (e.g., cost) and weight. Thus, one must evaluate the sum of the cost and weight for all indexes yielding 1 in the binary solution vector to find the greatest cost (for a maximisation problem) which does not exceed the weight limit. Other possible binary encodings can be used to encode non-binary information; for example, one could encode integers as 4-bit binary, i.e., 1010 would represent 10 in binary. These representations and operators need to be valid in the context of the problem.

Evolution Strategy

An evolutionary strategy (ES) is effective when an optimisation problem is best encoded as real number genotype values. ES candidate solutions are encoded as a real value vector $X = (x_1, \ldots, x_n) \in \mathbb{R}^n$ of length n. The vector index often represents the parameter, and the array value represents the parameter value.

Two of the most primal and prevalent EAs are the $(\mu + \lambda)$-Evolutionary Strategy and the (μ, λ)-Evolutionary Strategy. The number of parents in a population at each generation is denoted μ, and the number of offspring produced at each generation is denoted λ. For example, in the $(1+1)$-Evolutionary Strategy, a single parent and a single offspring exists in each generation. The $(\mu + \lambda)$–ES merge the parent and offspring before selection, whilst the (μ, λ)–ES means all parents are discarded at selection (non-elitist), although copies could exist as offspring.

2.2.5 Multi-Objective Evolutionary Algorithms

The field of Multi-Objective Evolutionary Algorithm (MOEA) development has been ubiquitous with the regular introduction of novel MOEAs (Sörensen 2015) (besides the point being made, the citation also debates the novelty of some recent MOEAs). Many of these novel developments are application-specific modifications. However, a handful of the more general use MOEAs (NSGA-II - Deb, Pratap, Agarwal & Meyarivan (2002), SPEA2 - Zitzler et al. (2001) and MOEA/D - Zhang & Li (2007), to name a few) have been highly popular within the field and are still used in modern EA research. Often these MOEAs have been considered as benchmark EAs. The most popular of these algorithms is NSGA-II.

In 1994, the Non-dominated Sorting Genetic Algorithm (NSGA) (Deb et al. 1995) brought advancements to the standard GA by implementing non-dominated sorting to achieve better results than traditional GAs on the same problems. NSGA-II was a further advancement of NSGA. NSGA-II brought three considerable benefits to the initial MOEA. The first was that NSGA reduced the computational complexity from $O(MN^3)$ to $O(MN^2)$ where M is the number of objectives and N is the size of the population. The second revision introduced elitism, and the third advantage was reducing the number of parameters required (removing the sharing parameter). These were done by introducing elitism (combining the parent and offspring population) and crowding distance into the selection operator. These adaptations resulted in improved results of NSGA-II from NSGA on the same benchmark problems for many-objective problems.

A key advantage to NSGA-II is obtained in the selection process of the algorithm, during which parent and offspring populations are pooled together to increase the selection pool size (increasing the likelihood of finding better solutions compared to using an offspring-only pool). Then non-dominated sorting occurs to rank the population and choose a predefined number of solutions, population size N, with the greatest ranks. However, in many cases, the cumulative number of solutions in a rank can exceed the population size, so a requirement for tournament selection for solutions with the same rank is needed - this is the role of the crowding distance operator.

The crowding distance operator is a calculation to obtain a density estimation metric for each solution and is used to compare and choose solutions for the population. The crowding distance considers solution (i) and its two closest neighbouring solutions adjacent to the solution, denoted $(i-1)$ and $(i+1)$. The crowding distance is a measure of the distance between these two solutions. Solutions with a lower crowding distance are closer together and hence not chosen over solutions with a greater crowding distance. This is in order to select those solutions in less crowded areas and provide the population with greater diversity. Furthermore, it does not require additional parameters to be set such as a sharing parameter, unlike NSGA.

Whilst NSGA-II is the more omnipresent MOEA found in the literature (38600+ citations), we shall briefly summarise two other algorithms used in later chapters. The first is Strength Pareto Evolutionary Algorithm 2 (SPEA2) with (8500+ citations). Similarly, for SPEA2 and NSGA-II the '2' or 'II' indicates an improved second-generation algorithm version. SPEA2 also has three major advancements. First, the improved fitness assignment, as previous SPEA (Zitzler & Thiele 1999) solutions that are dominated

by the same archive solutions yields the same fitness values, causing all solutions to have the same rank if only a single solution is in the archive. The second upgrade was to use a nearest neighbour density estimator (SPEA2 incorporates density information into fitness assignment, unlike SPEA, which used archive clustering) and, finally, a new archive truncation method was introduced.

Another popular algorithm is the Multi-Objective Evolutionary Algorithm with Decomposition: MOEA/D (6100+ citations) (Zhang & Li 2007). The MOEA/D's primary feature is to decompose many-objective optimisation problems into several subproblems (e.g., scalar aggregation function) and optimise simultaneously. The subproblems generate solutions, and future populations are comprised of the best solution for each subproblem. The MOEA/D with the Tchebycheff decomposition approach produced similar results to NSGA-II on benchmark problems (Zhang & Li 2007).

2.2.6 Many-Objective Evolutionary Algorithms

Many-objective problems exacerbate the difficulty for evolutionary algorithms in evolving optimal solutions. Often, EAs that were primarily designed for multi-objective optimisation problems become increasingly unsuitable as the number of objectives increases beyond three (Köppen & Yoshida 2007). For example, as the number of objectives increases, the search space expands exponentially. Given the solutions are uniformly distributed along the objectives the number of non-dominating solutions increases by the formula $1-(1/(2^{M-1}))$ as the number of objective M increases (Farina & Amato 2002). Furthermore, with many-objective problems, the dominance relation inadequately scales as there are exponentially more dominant solutions (Farina & Amato 2003). The functionality of diversity mechanisms like crowding distance deteriorates as the crowding distance computation increases. Diversity can be diluted with the exponentially greater cardinality of the dominated solution set, filling the majority (potentially all) of the population.

One of the most noteworthy many-objective EAs is NSGA-III (Deb & Jain 2013). NSGA-III is a successor to NSGA/NSGA-II. NSGA-III replaces the crowding distance operator with a reference plane based procedure. NSGA-III uses a similar framework to NSGA-II. However, a predetermined set of reference points is supplied to the EA, and the selection emphasises solutions closest to these points. This can increase diversity and overcome the aforementioned issues associated with NSGA-II. For example, uniformly spacing reference points creates a hyperplane across the different problem objectives, which can capture a better population representation of the search space. This reference plane process also has a lower computational complexity that the crowding distance (Deb & Jain 2013).

Often in the implementation, if predefined reference points are not known for the problem, we can supply a total number of divisions in each objective plane and denote this value p. Then, a normalised hyper-plane, a dimensional unit simplex (insects at one on each of the axes), can be created from that hyper-parameter. The number of reference points H can be calculated by

$$H = \frac{M+p-1}{p} \tag{2.10}$$

for an $M-$objective problem (Deb & Jain 2013).

NSGA-III provided results similar to MOEA/D when benchmarked (Deb & Jain 2013), which could also be used for many-objective problems as MOEA/D uses multi-predefined search directions (subproblems) to find widely distributed solutions.

2.2.7 Test Problems

A series of benchmarks need to exist to provide a standard for which EAs can be compared. As with evolutionary multi-objective (EMO) algorithms, a plethora of optimisation test problems exist. This section considers some of the most significant literature. A range of both discrete and continuous problems are required. Discrete problems such as the multi-objective 0/1 knapsack problem (Zitzler & Thiele 1999), NK-landscapes (Ochoa et al. 2008), the flow shop scheduling (Ishibuchi et al. 2003) and the travelling salesman problems for multi-objective optimisation (Corne & Knowles 2007) are commonly used as benchmark permutation problems. As well as dynamic problems for which the fitness landscape adapts over time, a survey of these problems can be found in the work of Jiang et al. (2022). However, a much greater range of continuous problems exist in the literature, and we shall consider some of these in this work.

One of the most omnipresent test problems in the literature is the Zitzler-Deb-Thiele (ZDT) problem set (Zitzler et al. 2000). The problem set comprises six test problems, each containing a feature known to increase problem difficulty. With regard to optimiser convergence, the problems contained features such as multi-modality, deceptive fronts and isolated optima to increase the difficulty. Further, convexity, nonconvexity, discreteness and nonuniformity features were included in the problems to create challenges for solutions converging and diversifying over a front. All problems contain at least one of these landscape features, with the final test problem possessing two difficulties. The problems were all dual-objective. This work delivered an effective way of examining algorithm performance using test problem benchmarks. However, later these test functions were criticised for being too simplistic or unrepresentative of real-world problems (Huband et al. 2005, Tiwari et al. 2002).

As EMO algorithms developed, the requirement for more challenging problems grew, leading to the development of the DTLZ problem suite (Deb, Thiele, Laumanns & Zitzler 2002). These test problems are the primary benchmark problems in EMO optimisation. The problem suite contains nine problems that scale to have any number of decision and objective variables, which significantly increases the problem difficulty when solving with EAs. The number of decision variables suggested in the work is $D = k + m - 1$, where $m - 1$ is the number of position parameters and k is the number of distance parameters, controlling the distance the solutions are from the Pareto front. A suggested k value is supplied for each problem in the work. Improved iterations of the DTLZ test suite have also been proposed more recently, such as the C-DTLZ test suite (Deb & Jain 2013), which consists of constrained problems to add additional challenges. The BBOB problem suite is another popular continuous test problem suite Tušar et al. (2016).

A more challenging test suite was later introduced, known as the Walking Fish Group (WFG) (Huband et al. 2006) problem suite. This work provided a rigorous analysis of existing test problem literature. Then propose a toolkit of transformation functions able

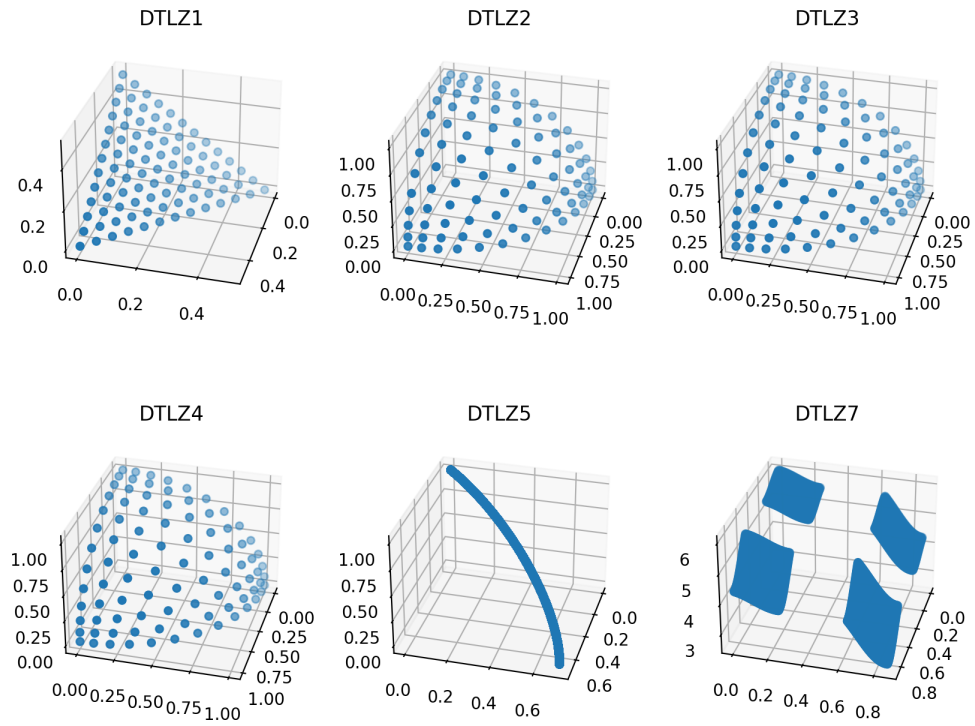


Figure 2.2: The DTLZ test suite problems utilised in this thesis.

to generate test problems with greater control of problem features than previous sets. The work then builds a series of nine problems known as the WFG problem suite to show the toolkit's ability. The work of Fieldsend et al. (2021), like the WFG problem suite, aims to create a test problem generator for which the designer can modify the features and the number of features.

As most benchmark problems consist of artificial mathematical functions, the works of (van der Blom et al. 2020) question the representation of benchmark problems for real-world problems. The work uses a questionnaire for experts and practitioners of optimisation to examine the types and frequency of real-world problem properties. The work finds features previously unconsidered in existing benchmarks, such as 44% of the real-world problems sampled require less than one second of evaluation time. Furthermore, providing an overview of key problem features and frequencies of problem types (i.e., the number of objectives) in real-world problems. Real-World-Like Problems (RWLP) also exist in the literature, focusing on simulating real problems from various applications such as radar waveform design (Azami et al. 2012), wind farm optimisation (Mosetti et al. 1994), car side impact (Dhiman & Kaur 2019) and water distribution network optimisation (Walker & Craven 2018*a*). Some conference competitions also propose problem suites used for benchmarking novel EAs, such as the CEC 2018 (Cheng et al. 2018) and CEC 2022 problems (Luo et al. 2022).

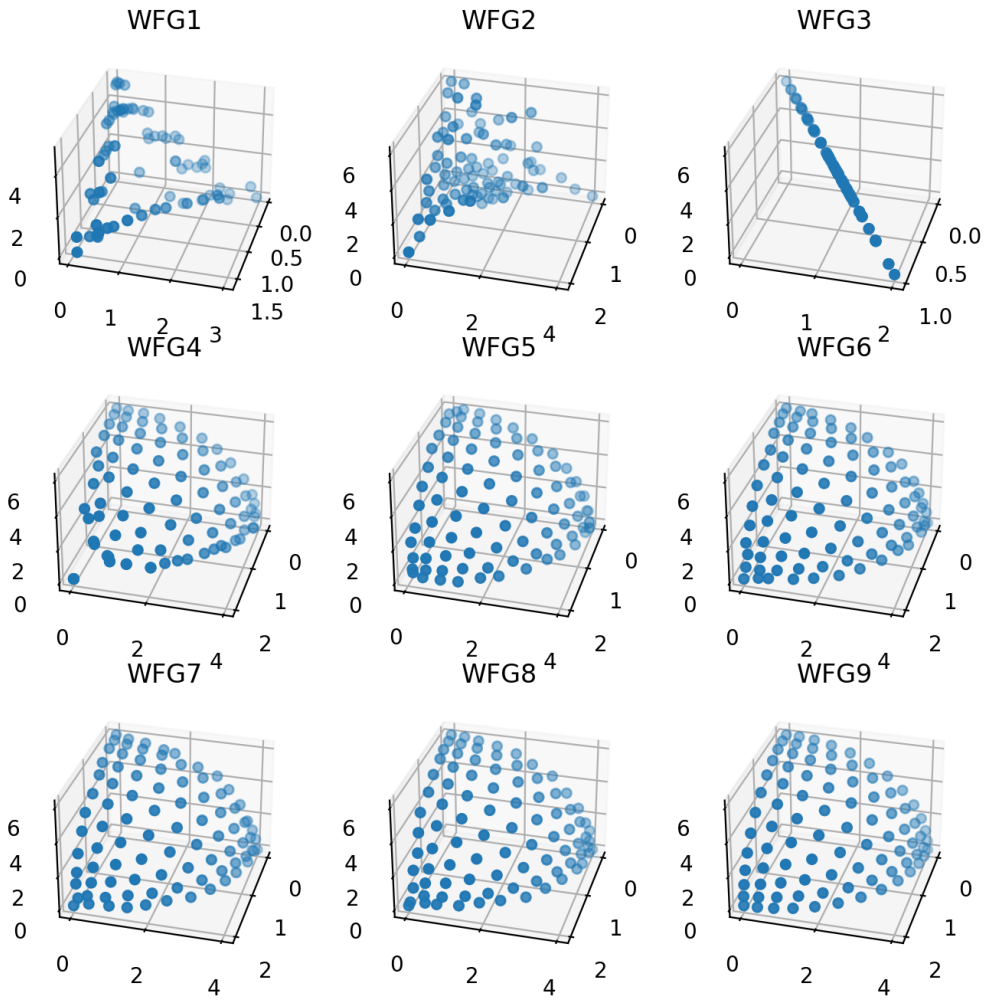


Figure 2.3: The WFG problem suite.

In the later chapters, explainable AI is a primary theme, and so we look at the existing methods of explainability and the benefits explainability brings to AI in the next Section. This allows one to examine the current explainability literature, consider the requirements for explainability methods and the effect explainability could have on evolutionary computing.

2.3 Explainable Artificial Intelligence

It is evident from the literature that *subsymbolic AI* (*neurological AI which considers intelligence using neural-networks*) is omnipresent and the dominant AI type of modern times (Cambria et al. 2020). However, symbolic AI also appears amongst the literature. Symbolic AI required rules defined by experts; it used human-readable rules, making it understandable and explainable but suffered from poor scalability. Subsymbolic AI was quite the opposite; it did not require human rulesets and was far more scalable.

However, just as we understand little about neurology and the function of the human brain, subsymbolic AI was also difficult to understand for a human user. Pre-1950 *symbolic AI*, which used symbols and expert logic, was the primary driver of AI and was believed to have held the most hope for intelligent systems at the time. In the 1950s, Frank Rosenblatt derived the 'perceptron' which set in motion the prospect of neurological AI or subsymbolic AI. In the 1960's, Minsky wrote a paper mathematically proving the limitations of subsymbolic AI in which they speculate on the "sterility" of MLP (Minsky & Papert 1969). That led to funding drying up for subsymbolic AI, and it was not until a series of AI springs followed by AI winters in the 1970s and 1980s, caused by an overhype and unrealistic expectations of over-optimistic scientists, marketing and poor scalability, that DARPA and UK research institute's in 1988 began to reinvest in subsymbolic AI. By this time, many of the limitations had been mitigated with the likes of new technologies such as backpropagation. Following on, both scientists and 'big tech' companies brought a technological revolution with aid from the introduction of mid-2000s deep learning. Consequently, there's been exponential progress with AlexNet (Krizhevsky et al. 2012), GoogLeNet (Szegedy et al. 2015) and GANs (Goodfellow et al. 2014), OpenAI's GPT-3 (Brown et al. 2020), AlphaGo (Silver et al. 2016), and significant progress in natural language processing and computer vision.

AI is a pillar of the 4th industrial revolution, and much of the future technological advances are dependent on it. AI is already used in many applications with decisions that affect people's lives, unfortunately yielding a black-box nature. This black-box AI brings into question many ethical, technical and legal problems around understanding and trusting AI, as poor interpretation of mission-critical AI can lead to severe consequences and even fatalities (Caruana et al. 2015). XAI is a priority on the research agenda of governments around the world as challenges relating to interpretability are a barrier to all; they recognise the significant benefits AI will bring to the economy, healthcare, and security whilst recognising the importance of explainable systems for successful integration. Explainable AI is required from a legal and ethical standpoint, raising questions about AI's ethical use (particularly in healthcare, justice and defence) and a lack of legal compliance with laws such as the European GDPR legislation setting out the 'right of explanation' (Goodman & Flaxman 2017) which requires systems that make automated decisions to produce explanations of the decisions and inference of the system which the subject can challenge.

Explainability can bring technological advances with less bias, more robustness, greater understanding of system limitations and more causality, i.e., if one can understand a system, then improvements are more obtainable. AI is increasingly relied upon to make more significant decisions in people's lives, and as AI becomes more automated (in its design) and complex, explainability becomes more and more difficult. As the AI research community has begun to recognise the importance of explainability for subsymbolic AI systems, there has been an exponential increase in publications relating to explainability within AI. Figure 2.4 shows this increase in publications for XAI. Many of these articles consider XAI in specific domains; others include taxonomies of the field. As a result, we see the distribution of literature negatively (left) skewed with many recent advancements, highlighting the nascence of explainable AI.

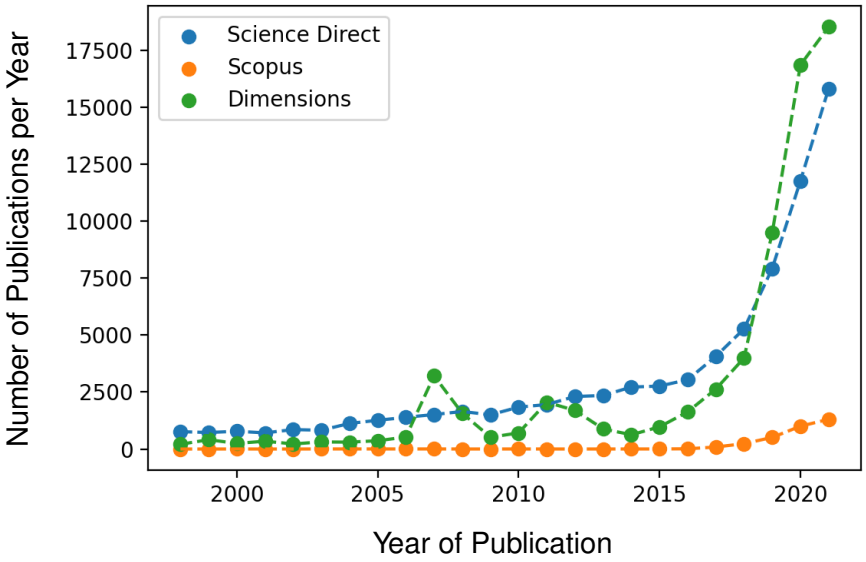


Figure 2.4: Database entries of the number of publications containing the term 'explainable artificial intelligence'.

2.3.1 Definitions and Prior Considerations

We have previously set out the motivation and importance of explainable and understandable AI. To create explainability methods, one first needs to define the terms and herein lies the first significant issue with XAI. What does explainable AI mean? This is a question that many scholars have attempted to answer, but there is yet to be a single widely accepted definition of XAI, even though many papers claim explainability. The challenges of explainability can be subjective and differ from application to application. For example, an AI scientist may require a different type, level, and accuracy of explainability than an AI user, such as a lawyer or doctor. The cognitive ability and psychology of the person may also require different types of explanation, so it is often audience-dependent. We note XAI is a multi-domain work focusing on social science, psychology, AI, philosophy, technical, legal and ethical domains.

There are many definitions of XAI; most of these consider an audience and a level of understanding of AI by the audience. A common misconception is that explainable AI only considers the interpretability of the model; however, XAI consists of several requirements dependent upon the audience and application. For example, from the survey work of Arrieta et al. (2020), explainability is broken down into nine subterms, with the degree of importance being application/audience dependent. These are as follows;

Trustworthiness is the confidence that the model will perform as intended on a problem. Unfortunately, trustworthiness is difficult to measure, and it can be the case that a model can be trustworthy and not explainable. Although greater explainability is likely

to correlate with more trust in the system.

Causality is the ability to determine relationships within the data features. For example, correlations between input features such as living closer to the beach or the size of the property may affect the house price-output prediction from a linear regression model. These causal feature relationships should be clear and preferably measurable in an explainable system.

Transferability is a trait of the ML model needed to reuse existing ML models on other problems or make adaptations to the model. For transferability, we require an understanding of model limitations. Model generalisation somewhat supports transferability to some degree.

Informativeness considers that the problem being solved by the ML model is not the same as the problem solved by the human. ML models should be informative about the true problem and the workings of the ML model solving the problem to reveal misconceptions. This could consider the Rashomon effect (Breiman 2001) (coined from the 1950s film 'Rashomon' where four characters describe different perspectives of the same crime) which states there can be many different explanations for the same phenomenon, i.e., one model may use one feature primarily to make a decision and another model may use another feature to make the same output.

Confidence aims to depict model robustness and stability. ML models should reveal a level of confidence about their inner workings, and this trait, therefore, ties closely with trustworthiness.

Fairness is concerned with providing models that are not unethical and do not negatively impact people in an unjust way. For example, this could be caused by an event such as data bias. Fairness, therefore, needs to consider the social impact of the model.

Accessibility allows more people potentially of nontechnical backgrounds to develop and work with ML. This term is highly correlated with the complexity of the model's explanation.

Interactivity allows users to interrogate the ML model in an interactive way. For example, interactivity could enable the user to focus on specific areas that require more explainability for that user or ask the explainability model specific questions.

We then consider *privacy awareness*. If one can not understand the patterns captured in the model's data (Castelvecchi 2016), which has been trained using real-world datasets, there could be a breach of privacy and privacy laws. Moreover, training a model using real-world data on people means the model weights are representative of the real-world data and thus bringing into dispute ownership of the model. Questions also arise about the possibility of reverse engineering the models to obtain sensitive data. This relates to model security; whilst not a subterm in the cited section work; we would like to consider this addition to the benefits obtained from greater ML explainability. Finally, whilst the survey work categorised the nomenclature, it should be noted that significant overlap exists between the terms.

Other work (Lipton 2018) also breaks down the levels of transparency an XAI system should entail. These are the notions of *simulatability*, which considers how much

depth a human can think about the inner workings of the systems. For example, a human can envision a single perceptron working, but multiple perceptron layers are more complicated, with too many connections and neurons to consider. As the number of neurons or rules increases, the model becomes less simulatable. Next, *decomposability* considers to what level an AI system can be decomposed into its constituent components (i.e., input, parameters and output). Finally, it considers *algorithmic transparency*, which considers the degree of confidence a model will perform rationally given an input.

As well as the need for an agreed unified definition for explainable AI, there also exists no single agreed upon metric for measuring explainability in a quantitive way. The work of Arrieta et al. (2020) advocates a requirement for a metric that can evaluate the performance of XAI techniques. For now, the convention is to use well established scientific metrics such as goodness of fit scores (R - value) or accuracy metrics such as F_1, recall, precision and accuracy. The work of Hoffman et al. (2018) attempts to establish some XAI metrics, such as the explainable satisfaction scale, but also notes far more work is required to quantify XAI. We should also consider the domain and audience of these terms given the problem the model is attempting to solve and the real-world problem is not identical: for example, the model causality is not the same as real-world problem causality, and this can be demonstrated with adversarial examples.

Explanations can further be divided on the scale to which they provide an explanation of the overall system. Some methods are considered *global methods*, which provide an explanation of the overall system, for example, an explanation revealing the internal mechanics of a prediction model for all instances. In contrast, *local methods* provide explanations at the individual solution level, considering an explanation for an individual instance. Some methods, such as LIME (Ribeiro et al. 2016) and SHAP (Lundberg & Lee 2017), can be used as a global or local method (usually by aggregating local values of individual predictions).

We also note many organisations adopting AI are now beginning to adopt and incorporate XAI principles within their organisations. Most principles focus on responsible AI, which is fair, accountable, ethical, and considers privacy and security. These principles are derived to make the company more ethically sound and create trust amongst clientele. However, there exist many anomalous examples of AI misfunctioning to cause unexpected and sometimes shocking outcomes, so these principles are often updated. For example, Google demonstrated Google Assistant, which was able to mimic a human over a telephone booking a table at a restaurant; surprisingly to Google, the level of anthropomorphism was seen by some as controversial and upsetting, which led Google to add a clause to the principle requiring AI to first identify itself as AI before human interaction (O'Leary 2019).

2.3.2 Directly Transparent Models

Not all AI models require additional explainability techniques. Many AI models exist that are inherently transparent and provide considerable levels of explainability out of the box. These models that are explainable by nature will be defined as *directly transparent models* and sometimes defined as *intrinsic explanations*. Unfortunately, there does not appear to be a single transparent model that yields the highest accuracy for all

applications of AI. A phenomenon for which the more explainable the model, the lower the accuracy for complex datasets. This is because explainable models are less complex, hence simple to understand at the cost of the model not capturing more complex data relationships.

Some directly transparent models include linear regression, which involves fitting a linear model (a gradient and intercept constant) to the data and making regression predictions with the linear model. Alternatively, a series of linear models can be used to model the data as Generalised Linear Models (GLM) (Nelder & Wedderburn 1972). The model is explainable as a stand-alone model; however, it can only capture linear variable relationships, which could obtain poor accuracy for non-linear data structures. From both the model and a graph of the model (most explainable models are still complemented with visual representations), one can determine the linear variable relationship and strength. Statistical tests can also be conducted on the model fit and model's predictors to determine how well the model predicts and hence represents or explains the data relationships.

Other methods under the umbrella of transparent models are decision trees, tree ensembles, random forests, gradient tree boosting such as XGBoost (Chen & Guestrin 2016), and other tree-based models, given their ability to yield human-readable rules. However, as the number of rules and the tree size increases, interpretability tends to decrease, and the models often are poor at generalisation. The same advantage and disadvantages apply to rule-based systems like fuzzy logic, where human-readable rules can be extracted. Bayesian models also exhibit a degree of explainability by considering the conditional dependency of variables and fitting a distribution to the data. However, the naive Bayes variable independence assumption is required from the data, requiring features to be independent.

The Generalised Additive Model (GAM) (Hastie 2017) uses a linear aggregation of smoothing functions to predict a single feature of the input dataset, thus, allowing one to quantify the impact of each input feature to provide some level of explainability. GAM is an explainable model which can capture non-linear variable relationships and provide flexibility but may not yield the accuracy of more complex black-box models. GAM is similar to GLM, except non-linear functions are fitted to the data as opposed to a linear function, and so non-linear relationships can be captured. Although typically a post-hoc method, Generalised Functional ANOVA functions can also be used to decompose the feature relationship function, usually into the intercept, baseline and feature interaction effects to explain the features and the inter-feature relationships.

2.3.3 Post-Hoc Explainability Techniques

Many AI models are not inherently explainable by design and, as such, exhibit *black-box* behaviour. Methods from the explainability literature need to be implemented after model training to produce additional explainability of the model and make these models more human-interpretable. These *post-hoc explainability techniques* can be applied to trained models to provide, as surveyed in Guidotti et al. (2018), text explanations, visual explanations, local explanations, explanations by example, explanations by simplification and feature relevance explanations to provide insight into the inner working of the model converting from *black-box* to *white-box*.

Text explanations provide interpretability via symbols; this could be in the form of human-readable pseudo-code or complete written language sentences. For the relevance of this work, we shall focus most on visual explanations; these methods aim to depict the algorithms' inner workings via visual representation, i.e., graphical representations. Visualisation is often a preferred method, given the human ability to recognise visual patterns quickly (Arrieta et al. 2020, Vilone & Longo 2020). Local explanations involve segregating the data into subsets that contain less complex, easier to explain data. The disadvantage of using a subset of the data for explanation is that more complex subsets are ignored and still not easily understandable. Explanations by example techniques focus on explainability by providing instances of the model working so that humans can theorise expected model outputs on given instances. Explanations by simplification involve methods that aim to simulate the model with a simpler simulation of the model whilst maintaining a degree of likeness to the model. Finally, feature relevance explanations aim to describe each input feature's impact and sensitivity on the model's predictivity.

The specific techniques can be categorised into *model specific*, for which the explainability method can only work on particular model types and *model agnostic techniques*, which are not model specific. Model agnostic techniques have, by definition, greater scalability for use on many model types.

Model Specific Techniques

This subsection will describe some model-specific techniques. These techniques only work on appropriate models; for that reason, we will only consider a few of the most common methods as much of this is not directly relevant to the following work in this thesis. The most common model-specific methods are for use with convolutional neural networks.

Many of these methods consider the gradients of a model. The gradient identifies how much of an effect one variable (e.g., each pixel in an image I) is having on another variable (e.g., probability of class Y_c). Class Activation Maps (CAM) (Zhou et al. 2016) insert a global average pooling layer (GAP) into a trained neural network (after the last convolution layer and before the final fully connected layer) to obtain information on the pixel's degree of effect on the model's output. The last layer is employed as it contains the best high-level semantics as an aggregation of filters, whilst the final fully connected layer only provides the probability of output. The GAP learns weights which are weights of the importance of the latter filters of the CNN, by training on the output of the CNN. One can then reconstruct a CAM heatmap by multiplying the weights by the filters and superimposing the CAM over the original image to show which pixels/regions have importance on the model prediction. This technique is demonstrated in Figure 2.5(f).

To use CAM, one needs to train a new model with the added GAP layer; this new explainable model is not the same model as the previous model, albeit similar. Moreover, the new model may yield a reduced accuracy than the original model for the trade in interpretability. To overcome this challenge, one can implement a variant of CAM defined Grad-CAM (Selvaraju et al. 2017), abolishing the need for the GAP layer and retraining by using existing network gradients. To implement Grad-CAM for class c, we first consider the gradients $\frac{\delta Y^c}{\delta A^K_{ij}}$ via backpropagation with respect to the K-feature maps of

convolution layer A and probability for target class c. This is calculated from the output class layer to the final convolution layer which determines the weights most significant to predicting the class. Then, the gradients are global average pooled to obtain the K-scalar weights (w_K^c captures the importance of feature map K for target class c):

$$w_K^c = \frac{1}{z} \sum_i \sum_j \frac{\partial Y^c}{\partial A_{ij}^K} \tag{2.11}$$

where i is the width dimension and j is the height dimension. We then utilise these importance weights to calculate the weighted sum of k filters of the last convolution layer to produce a heatmap superimposed over the input image being explained. Finally, we use the ReLU activation function to restrict the domain to positive values/impact on the pixels.

$$L_{Grad-CAM}^c = \text{ReLU}\left(\sum_K w_K^c A^K\right) \tag{2.12}$$

Many variants of Grad-CAM exist, such as Grad-CAM++ (Chattopadhay et al. 2018), which uses a weighted average of pixel gradients to give more importance to gradient weights as the original work assumes equal pixel gradient importance, which can suppress activation maps with a smaller spatial footprint.

Vanilla gradients (Simonyan et al. 2013), also known as Saliency maps, utilise a complex model's existing gradients to calculate the gradient of the loss function for a class with respect to the input pixels after a forward pass of the input image I.

$$\text{saliency} = \max_{r,g,b}\left(\left|\frac{\partial Y_c}{\partial I}\right|\right) \tag{2.13}$$

The output value of class c is denoted Y_c. Positive image gradients correspond to pixels with a positive effect on the class classification, and negative image gradients correspond to conflicting classes. Guided backpropagation (Springenberg et al. 2014) works similarly but zeros out the negative gradients to produce clearer results (guided as it considers guidance signals from higher levels during backpropagation). Similarly, other methods to create activation maps include:

DeepLift (Shrikumar et al. 2017) provides a way to overcome the discontinuity gradient problems and gradient saturation problems (i.e., ReLU limits activation below zero to zero) by backpropagating a reference (such as a baseline image) along with the image requiring explanation and computing the difference from the gradient of the image to the reference image which is then propagated through the network. SmoothGrad (Smilkov et al. 2017) adds Gaussian white noise to each pixel and, for many copies of the original image, then takes an average of the gradients to average/smooth the derivative fluctuations that occur when on a small scale.

Integrated gradients (Sundararajan et al. 2017) are also an attribution method requiring the gradient information from complex model layers to determine the feature importance of model predictions. First, a norm or baseline (e.g., completely black images or

noise) x' is required. Then, we integrate from the baseline to the instance X by considering the difference between the two images and summing/integrating the gradients between the differences in images. The evaluation of the prediction and integration stage is usually performed between uniform step sizes (the cited work suggests values between 20-300) from baseline to input $(0,1)$.

$$\mathrm{IG}_i(x) = \left(x_i - x_i'\right) \int_{\alpha=0}^{1} \frac{\partial F\left(x' + a \times (x - x')\right)}{\partial x_i} d\alpha \tag{2.14}$$

Concept Activation Vectors (CAV), also called TCAV, test the importance of certain features we, humans, hypothesise (concepts) to effect predictions (i.e., is the wolf or husky classifier model predicting a wolf because of background snow in the image?). To use the CAV method, a set of random images and images containing the feature of importance hypothesised (snow) are passed through the network to obtain this network's activation for these images. One then trains a linear classifier to separate the random images from concept images. The orthogonal of the linear boundary is the CAV; thus, this vector points towards the effect. We then use the gradient of the class probability (e.g., wolf classification score) with respect to the CAV for the concept (snow) to produce a CAV score for the concept. The higher the CAV score change for the concept, the greater the effect of the concept on the probability of classification for the class.

Model Agnostic Techniques

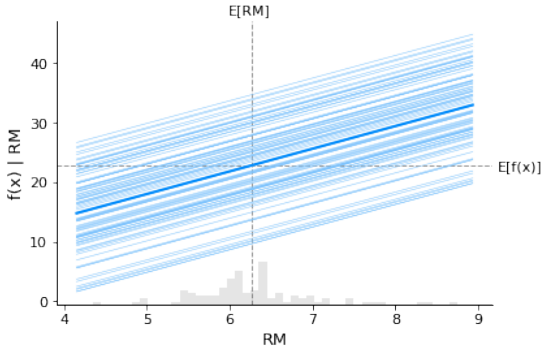
Perhaps the most obvious way to understand a model is with naive approaches such as removing a feature from the dataset and evaluating the new instances output to determine how that feature influenced the model. However, when features interact, this can create misleading results, potentially leading to unrepresentative inputs. So more rigorous methods are required to deal with these non-independence feature problems.

One of the most prevalent XAI methods is the Local Interpretable Model-agnostic Explanations, or LIME (Ribeiro et al. 2016), which as a local method, uses instances to create a local approximation model. It considers an individual instance and constructs a surrogate model (usually a linear model) to create a locally faithful simpler model (but not necessarily globally faithful). The model's universality is down to its scalability to any black-box model (it requires only inputs and outputs of the complex model), and it performs for many types of inputs (i.e., images and highlighted-text explanations). We also note that the evaluation metric of the surrogate model can be different from the black-box model metric, including having a higher metric than the black-box model - this is because the surrogate model trains using the black-box model outputs and not the original training data like the black-box model uses (this could reduce the problem complexity).

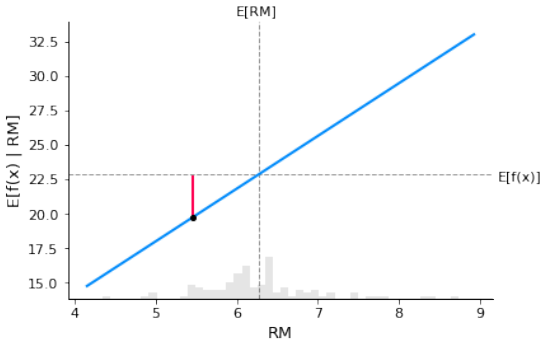
The model can be defined as

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \mathscr{L}\left(f, g, \pi_x\right) + \Omega(g), \tag{2.15}$$

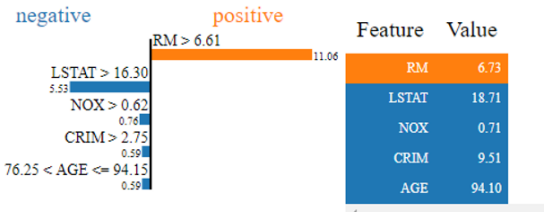
where x is an instance (input $feature_1, \ldots, feature_M$ for M number of features), g is the simpler surrogate model we intend to create, G is the set of explainable simpler models



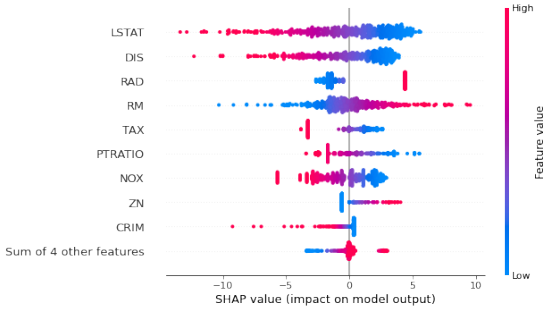
(a) ICE



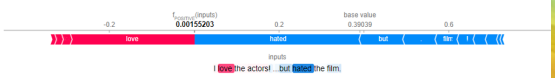
(b) PDP



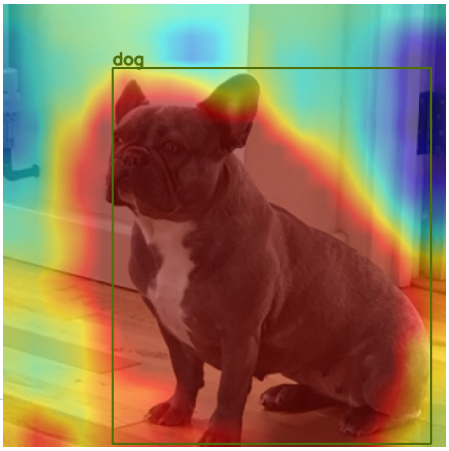
(c) LIME



(d) Beeswarm plot of SHAP values



(e) SHAP for NLP



(f) CAM

Figure 2.5: Common XAI methods demonstrated on tabular, text and visual inputs. Tabular dataset from the Sklearn Boston housing data. The pretrained model is YOLO5 (Jocher et al. 2021). The visual input is the author's dog.

(usually linear regression flavours), $\mathscr{L}$ is the loss function we aim to minimise, f is the original complex model we intend to explain, π_x is the neighbourhood of x or the proximity, $\Omega(g)$ is the regularise term of the model to simplify the model (i.e., for linear regression a zero weight could be used for some features).

In the previously cited work the loss function used was the sum of squares:

$$\mathscr{L}(f,g,\pi_x) = \sum_{z,z' \in \mathscr{Z}} \pi_x(z) \left(f(z) - g\left(z'\right)\right)^2 \tag{2.16}$$

with $f(z)$ being the instance truth, $g(z')$ is the prediction from the simple linear model of the instance, and the π_x is the proximity weighting.

To use LIME, we need to consider an instance of the dataset and create new neighbourhood data points by applying small perturbations, in the direction of features, to copies of the original instance; these perturbation sizes usually follow the normal distribution with the mean μ and standard deviation σ equating to the μ and σ of the features. The neighbouring points are weighted on the distance from the instance, usually using an exponential kernel; therefore, data points closer to the instance have more significance. The labels for this new dataset are generated by inputting the new dataset into the complex model.

We can create a simple linear regression model for the local instance with the new dataset, which is directly interpretable:

$$f(x) = g\left(x'\right) = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{2.17}$$

The linear regression constant can be displayed as a correlation chart with the feature ($feature_1, \ldots, feature_M$) the coefficient represents on one axis and the value of the coefficient (ϕ - the effect/correlation the feature has) on the other axis. Other representations include excluding words or pixels to generate the new samples and highlighting text or pixels of the features with the greatest effect. The original work considers using clusters of pixels called superpixels to create new samples for image representation.

Other methods that consider feature relevance, which could reveal important correlations between features, such as a model using a protected class (for example, age), would be the Shapley Additive exPlanations or SHAP methodology (Lundberg & Lee 2017). The use of Shapley values was inherited from the field of game theory. Shapley values would estimate each player's average contribution in a cooperative game to evaluate each participant's contribution distribution. For the use in AI, we consider each feature's estimated contribution for some instance on the complex model.

As inter-feature relationships exist, one can not simply remove a single feature from an instance and then evaluate the feature minus the instance to determine the feature's contribution. Therefore, we are required to evaluate many combinations of features for the instance; these combinations are called coalitions. Furthermore, in order to exclude features we wish to ignore, we cannot just remove the column, or it would not be the correct input shape for the complex model; instead, we fill the feature column

with random data or a background dataset to maintain the shape of the testing data and the random data should provide zero predictive power.

To calculate the SHAP value for feature i:

$$\phi_i(f,x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M-|z'|-1)!}{M!} \left[f_x\left(z'\right) - f_x\left(z' \mid i\right) \right]. \tag{2.18}$$

The first term provides a weighting, depending on how many features $|z'|$ out of the total number of features M are in the subset. We provide a greater weighting to the smallest and largest subsets as if we add a feature to a subset that already contains many features or few features; if the value changes significantly, that is an indication that this added feature has a significant impact (either single contribution or high inter-feature relationship contribution). All possible subsets of features for the instance are denoted x'. The second term $[f_x(z') - f_x(z' \mid i)]$ provides the marginal value, $f_x(z')$ determines how much each feature contributes to the subset by computing the instance on the complex model with feature i of interest, and $f_x(z' \mid i)$ evaluates the instance without the feature i of interest.

There are 2^M combinations of features for M features making this NP-hard; this equates to high computation complexity method. This led to other methods of computing SHAP, such as kernel SHAP (combining LIME and Shapely value methods), which uses samples of subsets to create a linear model for which the variables are either 1 or 0 depending on the presence of the feature in the subset and coefficients are the approximate Shapley values for the feature. Whilst KernelSHAP is model-agnostic, model-specific techniques also exist, such as MaxSHAP, DeepSHAP or TreeSHAP (which changes the complexity from exponential to polynomial (Lundberg et al. 2018)), which can explain deep neural networks or tree ensemble methods like random forests.

The original authors describe the need for local accuracy, missingness and consistency to be present in an additive feature method. The principle of local accuracy states if the input of the complex and simplified models are approximately the same, the output should be similar $g(x') \approx f(x)$ if $x' \approx x$. Meaningless requires the attribution to be zero if the feature is excluded from the dataset; therefore, the model can only be affected by inclusion, not the exclusion of features $x'_i = 0 \implies \phi_i = 0$. Consistency requires that if a feature affection predicts in the complex model, that same feature cannot affect the local model in an opposite way. The authors claim SHAP as possessing all three.

Partial Dependence Plots or PDP (Friedman 2001) can also be used to visualise the Shapley value. The feature under consideration is plotted on the x-axis and the partial dependence of the feature (output from the model after fixing a feature or SHAP value for feature) on output on the y-axis. Usually, the distribution of data for that feature is superimposed over the PDP as a histogram. PDP is model-agnostic as it only considers model inputs and outputs. It is mostly used for global explainability.

PDP works by considering a feature's importance or the importance of two features' effect on the model, i.e., It can show if the relationship between the prediction and feature value is linear or complex. We start by fixing the feature value of importance to become constant for all samples and keep every other feature for each sample in

the dataset the same, so all other features maintain the original test set values. We input the new samples into the model to obtain new outputs; then, we average over all outputs. One can then plot the constant against the new evaluated samples and do this for many constants. The partial dependence function can be formally defined as

$$\begin{aligned} f_S(x_S) &= \mathbb{E}_{X_{\bar{S}}}\left[f(x_S, X_{\bar{S}})\right] \\ &= \int f(x_S, X_{\bar{S}})\, d\mathbb{P}(X_{\bar{S}}) \\ &\approx \frac{1}{n}\sum_{i=1}^{n} f\left(x_S, x_{\bar{S}}^{(i)}\right), \end{aligned} \tag{2.19}$$

where x_S are the features of interest (plotted in the x-axis) and X_s are all other features in the dataset (the complement). As the probability function is often unknown, the final term (Monte Carlo method) is used to approximate. n is the number of samples in the dataset. PDP takes the assumption that the features x_s and X_S are independent. For example, suppose most of the samples fall into one of two polar classes, and we take an average. In that case, if features are correlated, the average of the data could be located in unrepresentative regions of the space. Accumulated Local Effects (ALE) (Apley & Zhu 2020) provides an extension to deal with this challenge. Furthermore, heterogeneous effects are not observable as an average is taken over all dataset samples. For example, if for a feature, half of the samples have a positive correlation with output and the other half have a negative correlation with output, and we take an average, we could get a result which would cancel the effects out, thus producing a horizontal line inferring no relationship, one solution is to use Individual Conditional Expectation or ICE (Goldstein et al. 2015) plots to reveal heterogeneous effect. In the visualisation, often, the samples are plotted as a histogram on the PDP graph to prevent over interpretation of the plot where the data distribution is low.

The ICE plot is analogous to PDP. However, it considers a single instance rather than an average of samples. So a visualisation would contain a line on the plot for every sample rather than a single average line like PDP. Extreme sample values are visible, which would solve the heterogeneity problem. For correlated features, we could use M-plots to consider conditional distribution to average over the realistic samples only that fall within the distribution (needs to be pre-calculated) so that unrealistic data is not considered. Alternatively, ALE plots divide the samples up into feature bins and calculate the average distance each sample is to the two border bin divisions, which removes the effect of the other correlated feature by accumulating the gradients instead of the function value.

In a similar way that PDP provides a feature attribution analysis, sensitivity analysis can also provide information on feature sensitivity for instances inputted into the model. To calculate each feature's effect on the model, we can change a single feature for an instance and keep all other feature values constant to determine the sensitivity. Different transformations can be applied to the dataset to create instances with changed feature values, often sampling changes from either zeroing out, sampling from the uniform distribution or the normal distribution (to keep values similar to the dataset).

Counterfactuals provide an explanation for some desired request for the magnitude of

an instance to classification; for example, one may want to know what feature values are required to achieve the desired classification for some specific instance; therefore, the instance change should be minimal and achievable (e.g., can not change the sex of a person). We can formulate this as an optimisation problem to find the minimal change required to achieve classification. The work of Dandl et al. (2020) considers the counterfactual finding a multi-objective optimisation ($M = 4$) and solves it with an evolutionary algorithm (NSGA-II).

The work of Wachter et al. (2017) formulates the counterfactual problem and solves as:

$$\begin{aligned} \mathscr{L}\left(x, x', y', \lambda\right) &= \lambda \left(f\left(x'\right) - y'\right)^2 + d\left(x, x'\right) \\ d\left(x, x'\right) &= \sum_{j=1}^{p} \frac{\left|x_j - x'_j\right|}{MAD_j} \\ MAD_j &= \text{med}_{i \in \{1,\ldots,n\}} \left(\left|x_{i,j} - \text{med}_{I \in \{1,\ldots,n\}} \left(x_{1,j}\right)\right|\right) \end{aligned} \tag{2.20}$$

The first term, $\lambda \left(f\left(x'\right) - y'\right)^2$, considers the difference between the desired value and current value with the λ providing a weighting to enforce importance to the first term, the addition term $d(x, x^{'})$ is the distance of the current instance from the original instance. The MAD term is the Mean Absolute Deviation. Other variants of counterfactuals include growing spheres Laugel et al. (2017) which sample a growing neighbourhood around an instance with an increasing radius until the desired outcome is reached.

In a similar way to counterfactuals, Learning Classifier Systems (LCS) (John 1976) are also ways in which evolutionary computing can generate interpretable rule sets for complex datasets. LCS generates a population of rules to describe complex ML datasets, using genetic operators such as mutation and crossover to replace/evolve components of the rules. The fitness of the rule is determined by how many data points fit with the rule's conditions, so more matches result in a greater fitness. The rules evolved need to be interpretable and should take a standard form such as the sUpervised Classifier System (UCS) (Bernadó-Mansilla & Garrell-Guiu 2003) which uses the format 'IF conditions THEN action,' e.g., IF the house has a pool AND the house has a view THEN house price is high. A population of these rules can be generated that describes the patterns in the data with a high degree of interpretability. However, sometimes LCS create too many unhelpful rules. So work exists to mitigate this issue, such as combining LCS with a clustering algorithm to prune important features for rule generation (Andersen et al. 2022).

Genetic Rule EXtraction (G-REX) (Konig et al. 2008) is also a rule-based explanation using the input and output of the complex model to train on, employing genetic programming to extract and generate IF-THEN rules from a dataset and model. First, the GP initiates a population of candidate rules (using the ramped half and half strategy), i.e., the function sets such as arithmetic operators and terminal sets such as input variables and constants, to define the representation language. Some solutions are chosen with roulette wheel selection and perturbed (random combinations of terminal and functions sets) using crossover and mutation before being evaluated on objec-

tives which are the conflicting accuracy (making the same classification as the complex model) and comprehensibility (length of rules set program). Finally, a syntactic validity check is done throughout the process.

The Conjunctive Normal Forms (CNF) or Disjunctive Normal Forms (DNF) trees (Su et al. 2015) are other examples of classification rule extraction methods. To do so, we create a decision tree (directed acyclic graph) with the nodes of the tree representing the propositional variables (e.g., q = 'age < 30'). The decision tree represents the propositional connectives as a conjunction $\wedge$ (and) and disjunction $\vee$ (or) of variables. We aim to optimise this tree, usually against the conflicting objectives of complexity vs accuracy, with some optimisation solver. Again, this tree requires the input and output data to train and optimise. In the next section, we consider explainability for evolutionary computing rather than explainability for machine learning.

2.3.4 Explainable Evolutionary Computing

While the field of XAI is nascent, with most works only published in the last few years, the field of Evolutionary Computing Explainable Artificial Intelligence (ECXAI) is almost nonexistent, with the phrase ECXAI coined for the first at the Genetic and Evolutionary Computation Conference (GECCO) ECXAI workshop held in July 2022. One could argue that an elementary level of explainability has already been obtained with the current EC work, such as the visualisation of non-dominated solutions. However, the degree to which this is explainable and creates an understanding of the algorithm's inner mechanics is shallow, e.g., a single unadapted heatmap cannot answer questions like "How was a solution created?", "How do the mutation and crossover operators affect the search?" and "What are the solution's parents?". So ultimately, a level of explainability already exists within EC; however, the degree of explainability is shallow compared to the insight XAI methods provide to ML and DL models, causing many of the challenges we describe in later chapters.

For this reason, not all visualisations, which usually provide some shallow level of explainability, can be classed as ECXAI methods, as there should be some level of explainability for the model's inner workings. However, arguably, the solution predictions for XAI (for ML) methods are more interested in the final decision with respect to the model input (such as what features are impacting the model rather than the evolution of solutions and traceability, as this is not applicable to ML). In contrast, for EC, we have access to objective functions, which means we do not require a great degree of explainability of the final solutions alone. But, more challengingly, we are interested in the process for which those solutions are generated by the model so that we know the model is working in its intended way, the model is non-black-box (trustworthy), and we can provide solution traceability for checking robustness, supporting regulation and creating a deeper understanding of the algorithm and the problem landscape.

Not all EC needs explainable AI methods for interpretability; for example, genetic programming (GP), which outputs an interpretable decision tree, could be considered inherently/directly explainable depending on the tree size. However, other EC methods, such as EAs, are more difficult to interpret due to their representation combined with the stochastic processes, high-dimensional data and complex interactions of solutions with reproduction operators. Some of the first work to motivate ECXAI was that of

Fyvie et al. (2021). This work uses PCA to reduce the dimensions of solutions for two different metaheuristics; the centroid means of the solutions are plotted, and inference of the search trajectory is gained through various proposed metrics and statistics. The work is similar to Search Trajectory Networks (STNs) but in a non-graphical format and also uses centroids (STNs use nodes with the same effect) to allow for greater scalability of the method to a greater number of solutions. However, the increase in scalability is at the cost of losing the information at the individual solution level.

The work of Singh et al. (2022) considers the explainability issues for metaheuristics and devises a method to mitigate some of the raised issues. The method considers the effect/sensitivity of the variables of a solution. They propose a method which uses a support vector machine to build a surrogate (for use in the absence of time) to approximate the fitness of a solution. The solutions variables are mutated/altered and remeasured by inputting into the surrogate function to measure the effect/change of each variable with the addition of the parameter alteration. They then plot the values on a positive/negative bar chart with the variable on the x-axis and the contribution to the surrogate fitness on the y-axis. The work is performed on binary-encoded benchmark problems.

In the next section, we consider the existing literature for the visualisation of evolutionary algorithms. Combined with the EA and explainability literature, this section covers the basis of existing literature that the technical chapters later develop.

2.4 Visualisations for Evolutionary Algorithms

The maxim *a picture paints a thousand words* suggests visualisations can be more informative than a few words or a single metric as unlike single metrics, visualisation often provides a fast and intuitive way to find patterns and compare data, making them highly desirable for displaying large quantities of complex data. State-of-the-art EAs often produce large volumes of data such as the Pareto front or whole populations of solutions; therefore, the most meaningful way to show this substantial body of data would be visualisation. Many reasons exist for the desire to display data, such as algorithm interpretability, implicitly examining the convergence and diversity of solutions and providing an exploratory analysis of the solution landscape. Unfortunately, visualisation is not a panacea, and many difficulties arise for visualising large quantities of solutions and high-dimensional data types that EAs often produce. In addition, the best visualisation to use depends on the data being visualised. In a similar way that the *No-Free-Lunch theorem* precludes the existence of a single optimiser that can optimise all problems, no visualisation exists, which can best visualise all types of data (Gao et al. 2019).

The most common EA data to visualise is the approximation front generated by an EA. The visualisation taxonomy of Gao et al. (2019) segregates approximation set visualisations into three categories that we will use to classify the visualisation literature in this work. Other segregation methods exist in different taxonomies, such as the works of Filipič & Tušar (2018) and Tušar & Filipič (2014*a*). We also note the authors recognise some overlap between the three types of methods. The three types are direct methods, visualisation via a transformed coordinate system and visualisation via dimension reduction methods. However, visualisation of the approximation sets does little to de-

scribe the process and mechanisms for which solutions are evolved. The latter work in this chapter considers visualising the variation operators and whole populations, which usually is a more challenging task due to the greater numbers of solutions but at the reward of revealing much more informative visualisations to enhance algorithm interpretability.

In this chapter, a selection of Pareto front visualisations are provided in Figure 2.6. To generate an approximation set to visualise the example methods, we use NSGA-III with the outer division of the reference plane set at 12 reference points. The variators are Simulated Binary Crossover (SBX) (distribution index 15 and probability of 1) and Polynomial Mutation (PM) (distribution index 20 and probability of 1) operators. The algorithm evaluated the problem for 50000 function evaluations to generate a Pareto front cardinality of 456 solutions. The problem is the continuous DTLZ2 from the DTLZ test suite. The number of objectives is four, and the number of parameters is $D = k + M - 1$ where $k = 10$ and $M = 4$. The k variable is suggested in the original work and controls the distance the solutions are from the true Pareto front.

2.4.1 Direct Display Methods

Direct methods display the solutions in their full objectives. These methods are often the most intuitive, making them common in the EA literature but often scale ineffectively. They form some of the earliest visualisations, including scatter plots, some of the most prevalent visualisations in science. For single objective, single parameter problems the x-axis usually displays the parameter value while the y-axis displays the objective value. For greater than single objective solutions, we usually discard parameter information and plot the two objectives in the x and y axes. We can incorporate an additional third axis and colour or alter the shape of solution markers for higher-objective problems, but having more than a few objectives represented on a scatter plot becomes unmanageable, and numerous plots are required. The number of objectives M that can fit on a single 2-D plot is two, therefore as the number of objectives extends beyond three, a permutation of scatter plots can be used; this technique is known as the pairwise coordinate plot or a scatter matrix (Cleveland & McGill 1984) (illustrated in Figure 2.6(i)). However, as the number of objectives $M > 3$ grows, the number of required scatter plots to visualise all objectives very quickly becomes unmanageable $(M(M-1)/2)$; this phenomenon is known as *the curse of dimensionality*.

Parallel coordinate plots (PCP) (Inselberg & Dimsdale 1990, Fonseca & Fleming 1993) (illustrated in Figure 2.6(l)) are another way to directly display the objective values for each solution. The x-axis holds the objective number, the y-axis holds the objective value, and each coloured line represents a unique solution allowing, in theory, a large number of objectives and solutions to be visualised in their full set of objectives. However, as the number of objectives in a PCP plot increases, the visualisation becomes cluttered and challenging to interpret. Having a large number of solutions also decreases the clarity of the plot as differentiating between solutions becomes challenging. Both the pairwise scatter plots and parallel coordinate plots are the most commonly found visualisations in the EA literature.

Bubble charts were used in the work of Ashby (2000). Bubble charts were simple but could, at most, only visualise five-objective data. Bubble charts are similar to scatter

graphs with an objective value each in the x and y and z axes, but with the addition that a bubble represents each solution, the dimensions of the bubble and the colour of the bubble represent two additional dimensions.

Heatmaps (Eisen et al. 1998) are also a frequent method of visualising high-dimensional data, making them effective at visualising pairwise similarities for small populations (Pryke et al. 2007, Li et al. 2018, Hettenhausen et al. 2010) (illustrated in Figure 2.6(c)). Heatmap axes contain the objective number and the objective value is mapped by the colour of each cell. In contrast, visualising large population sizes (requiring many rows/columns) with heatmaps make population relationships difficult to observe, and limited information can be inferred about the population dynamics. Heatmaps also cannot show the structure of Pareto front. Later modifications were made by Walker, Everson & Fieldsend (2012) to use seriated heatmaps. Spectral seriation is used to re-order solutions and objectives in heatmap, enhancing the clarity of relationships within the high-dimensional data.

A more unusual approach to visualising solutions in high dimensions are Chernoff faces (Chernoff 1973). Chernoff faces display the values of high-dimensional data as features of a face, i.e., a mouth could represent the value in the first objective, the nose in a second objective, and the features properties reveal the magnitude of the objective value. For instance, the degree to which the face is smiling represents the goodness of the solution in objective one. The motivation behind the development of Chernoff faces was to capitalise on the human ability to quickly recognise facial features and small changes of expression. This method suffers from difficulties displaying large numbers of solutions in a practical way.

Distance and distribution charts (Ang et al. 2002) are used to plot the non-dominated solutions in the x-axis against the distribution in the y-axis to show how close each solution is to its nearest approximation front objective and the distance to other solutions. This visualisation aims to highlight solution density and hence the diversity of an approximation set. The visualisation displays the coverage of solutions on the front so issues can arise if the Pareto front is disconnected or poorly distributed.

Spider charts (García-Vico et al. 2019) (also known as radar charts and illustrated in Figure 2.6(g)) propose the normalised objectives in uniformly spaced directions, and the solution is represented as a coloured line reaching out to each objective by their magnitude. i.e., for a five-objective problem, each solution would create a pentagon shape with each of the five corners reaching out to reveal the magnitude of the solution at that objective. Multiple solutions can be plotted over the top of one another and identified by colouring to produce a visualisation similar to the structure of a spider's web. Unfortunately, these visualisations provide similar information and, hence, do not provide much benefit over the parallel coordinate charts.

2.4.2 Visualisation via a Transformed Coordinate System

Other visualisation methods are methods that transform the objective value into another coordinate representation. This includes radial coordinate visualisation (Hoffman et al. 1997) which is used to visualise the approximation front based on applying a non-linear mapping to represent the objectives as units on a 2-D circle. Radial coordinate visualisation preserves the approximation front distribution but not the geometry of the

front. The method also scales well to a large number of objectives. A common physics analogy for understanding radial-coordinate visualisations is to imagine a solution being connected to one end of a spring and an objective. As the solution is stronger in a particular objective, the springs pull tight and move the solution more towards that particular area of the circle. 3-D radial coordinate visualisation (Ibrahim et al. 2016) was later proposed, which added an additional dimension indicating the distance of the solution from the Pareto optimal front based on the hypervolume. 3-D RadVis (illustrated in Figure 2.6(a)) is effective at showing the distribution of the approximation front, such as convexity and concavity. RadViz was used in the work of Kitamura & Fukunaga (2021) to visualise a whole population with a focus on indirectly visualising problem constraints. The work utilised RadViz to plot the solution in the third dimension by the amount a constraint is violated (normalised), with colour (indicating time series) as a fourth dimension. Likewise, the polar coordinate system (He & Yen 2015) also maps high-dimensional Cartesian objective values to 2-D polar coordinates.

Another method that seeks to map the original 4-D approximation front into a new coordinate system is the prosection method (Tušar & Filipič 2014*a*). The prosection method (illustrated in Figure 2.6(f)) can visualise four-objective approximation fronts in three objectives whilst preserving many of the approximation front characteristics. It uses a prosection (projection of a section) to capture objective space information. Moreover, the method well maintains the Pareto dominance relations, shape, distribution and range of the solutions. However, it is challenging to scale for $m > 4$ objectives.

The prosection method requires the user to define a few variables such as the origin a, the $f_i f_j$ prosection plane, the angle φ and width d of the section. This forms the section and can be denoted in a tuple as $mD(a, f_i f_j, \varphi, d)$. Then all solutions in the section are projected from the f_i and f_j 2-D space into the combined $f_i f_j$ 1-D space. All solutions outside this section are ignored. The combined $f_i f_j$ dimensions can form an axis whilst the second axis can be an additional dimension. Hence the solution set dimensions are reduced by one by combining two objectives.

The star coordinate (Kandogan 2000) (illustrated in Figure 2.6(k)) and 3-D star coordinate method (Shaik & Yeasin 2006) project the Cartesian coordinates into a two- or three-dimensional radial space. All the objectives radiate from the centre of the circle/sphere; for each solution, objectives are placed uniformly around a circle, known as anchor points, with the objective component along the particular direction. A difference between Radviz is that points can lie outside of the circle for star coordinate plots.

The neighbouring relationships of solutions are maintained by preserving the relative distances between solutions. Later work, such as PaletteStarViz Talukder & Deb (2020), decomposes solutions into layers using star coordinates and then visualises these layers. The visualisation formed plots the layers as star coordinates with the x and y axes containing the objective anchors and the centroid distance in the z-axis stacking the layers.

The compass plot (Wang et al. 2021) is based on the Chinese navigation compass. The outer segment of the compass represents the hypervolume, and the inner segments represent the parameters. The main strength of the compass plot is to visualise the effect of parameters, as it does not show the Pareto front structure or reveal much about

the diversity (other than what can be obtained from the hypervolume) of the approximation set. However, challenges with dimensionality still exist, and the visualisation complexity could overwhelm the DM.

2.4.3 Visualisation via Dimension Reduction Methods

The final category used to classify approximation set visualisation is visualisation via dimension reduction. Many of the problems visualising EA solutions are due to the high-dimensional characteristics of the solution data. The curse of dimensionality refers to the phenomena associated with high-dimensional space that does not occur in more natural three dimensions, such as exponentially increasing combinations of order. In contrast, the manifold hypothesis suggests many high-dimensional data can lie on low-dimensional manifolds allowing one to implement a dimension reduction on high-dimensional data to create a lower-dimensional representation. Dimensional reduction visualisations can be an effective way to visualise the relationships and patterns in a lower-dimensional plane. However, reducing the dimensions means reducing the information within the data, so whilst dimension reduction techniques aim to reduce the dimensionality and maximise the original accuracy of the data, there is always a loss of information that could decrease the interpretability and usefulness of the data.

Some of the earliest work visualising the population in lower dimensions was the work of Pohlheim (1999), using predominately 2-D or 3-D line plots, and the work proposed implementing MDS for EA population visualisation (illustrated in Figure 2.6(d)).

Hyper-radial visualisation (Chiu & Bloebaum 2010) reduces the high-dimensional solution coordinates into two dimensions for plotting by grouping objectives and taking radial magnitudes of solutions in their grouped objective spaces. First, the objective values are normalised in case of different magnitudes of scales. Then objectives are grouped together into two groups; the hyper-radius (magnitude of solution from origin) of each solution is calculated for each of the two groups and again normalised to put the axes in the range $[0,1]$. This makes up the hyper-radial calculation (HRC); see equation 2.21. The two HRC values for each solution can be plotted, reducing these high-dimensional solutions to two dimensions. The HRC equations:

$$HRC = \sqrt{\frac{\sum_{i=1}^{m} \overline{F_1}^2}{m}} \tag{2.21}$$

$$HRC(1) = \sqrt{\frac{\sum_{i=1}^{s} \overline{F_i}^2}{s}} \tag{2.22}$$

$$HRC(2) = \sqrt{\frac{\sum_{i=s+1}^{m} \overline{F_i}^2}{m-s}} \tag{2.23}$$

Where HRC $\in [0,1]$ and $\overline{F_1}$ is the normalised hyper-radius of the grouped objectives. $HRC(1)$ and $HRC(2)$ are plotted.

Later visualisations included Self Organising Maps (SOM) (Obayashi & Sasaki 2003) from the clustering analysis literature. SOM (illustrated in Figure 2.6(j)) used an unsupervised competitive learning neural network to cluster high-dimensional data. It

provided a topology preserving mapping whilst projecting the approximation front into a lower (usually two) dimensional lattice.

SOM requires parameters (number of neurons, epochs and a learning rate) and the solution data. To use SOM we initialise SOM weights. We then choose a random solution and find its nearest Euclidean distance neuron (known as the winning neuron). The winning neuron and other neurons are then moved towards the solution (as the neurons are all connected), with the winning neuron moving the largest amount. This updates the Kohonen layer's weights (positions of neurons) with the weight update formula made up of the learning rate and topological neighbourhood; thus, neurons further away from the solution will have smaller changes in weights.

We repeat for all solutions and for a number of epochs. The result is all solutions are classified to some neuron, creating a classification where nearby points in the objective space are mapped to nearby neurons. The neurons are often coloured based on a unified distance matrix, with the colours representing distances between different neurons, i.e., lightly coloured neurons represent clusters of similar neurons and dark neurons are cluster boundaries. SOMs do not preserve the dominance relationship or the approximation front's geometry. Although scalable to many objectives, SOMs were challenging to interpret (Witowski et al. 2009).

The Generative Topographic Mapping GTM (Bishop et al. 1998) (illustrated in Figure 2.6(b)) was a constrained Gaussian probabilistic alternative of the SOM, in which model parameters are determined by maximum likelihood models, removing the need for a stress function and claiming to offer several advantages over the conventional SOM. GTM aims to project high-dimensional solution data into a low-dimensional latent space.

Neuroscale (Lowe & Tipping 1997, Everson & Fieldsend 2006) used initially as a non-linear topographic feature extraction method aims to preserve the inter-solution distances, also using a feed-forward neural network (known as a radial basis function) to create weights that minimise a stress function and then predict the coordinates of the solutions in the lower manifold space given the solution input. Like SOMs, we create a surrogate model (the neural network's weights) to develop a mapping between two spaces. Neuroscale has been shown to skew solution distribution after transformation for linear datasets, leading to poor results (Tušar & Filipič 2014*a*).

Sammon mapping (Sammon 1969, Valdés & Barton 2007) (illustrated in Figure 2.6(h)) aims to preserve the distances between solutions (minimising the stress function) from high-dimensional to low-dimensional space. Minimisation of the stress function is usually performed with gradient descent but can be performed with iterative methods. It is a non-linear method. The Sammon stress (error) to be minimised is formulated as

$$SS = \frac{1}{\sum_{i<j} d^*_{ij}} \sum_{i<j} \frac{\left(d^*_{ij} - d_{ij}\right)^2}{d^*_{ij}} \tag{2.24}$$

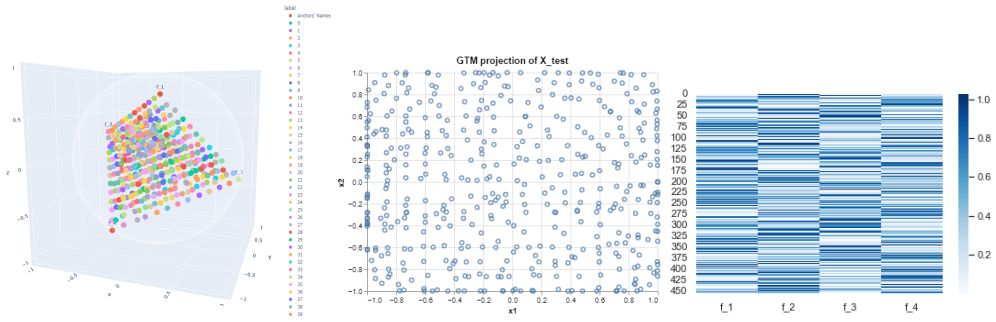
where d^*_{ij} is the distance between solutions i and j in the original space and d^*_{ij} is the distance between solutions i and j in the desired low-dimensional space.

Principle component analysis (PCA) from the machine learning domain was used to visualise the set of solutions to a scheduling problem (Masafumi et al. 2010). PCA (illustrated in Figure 2.6(e)) is used to detect linear structures within data. PCA seeks to preserve the maximum variance of data whilst projecting to a lower-dimensional space. The first principle component, or eigenvector of the data's covariance matrix, is the vector that preserves the maximum variance (eigenvalue). Multiple principal components can be formed by taking orthogonal projections of the previous principal components. Singular value decomposition of the data X is used to construct the eigenvectors and eigenvalues $X^T X = W\Sigma^2 W^T$, where Σ contains the eigenvalues in the diagonal $\lambda_1, \ldots, \lambda_k$ for the k largest eigenvalues required, and W is the eigenvector.

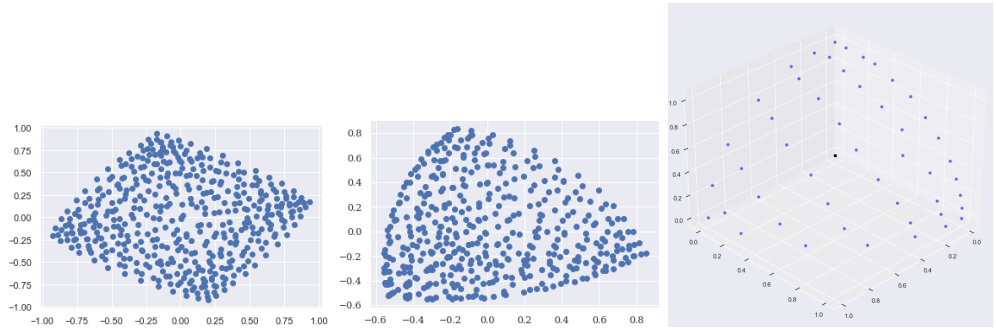
2.4.4 Visualising Populations

As shown from the examples in this chapter, sufficient literature exists to visualise approximation sets; however, the existing literature which aims to visualise the whole population of an optimiser is scarce. Unfortunately, visualising only the approximation set provides inadequate insight into the optimiser and the way the solutions were generated. In this subsection, we consider the insufficient literature, which does attempt to visualise whole populations. Whilst visualising populations can be informative, it carries many more additional problems than visualising approximation sets alone. Therefore many approximation set visualisation would not be suitable for visualising populations without significant adaptation. For example, a method that does not preserve the approximation front geometry will not likely preserve the population structure's geometry. Additionally, the number of solutions in a total population is greater than the approximation set and so high complexity visualisation methods which are unsuitable for large approximation fronts, are not likely to be suitable for visualising populations; this problem is likely to be exacerbated when applied to a greater number of solutions. Methods with high complexity would also not be suitable for visualising solutions generated as *a posteriori* methods where no time constraint exists.

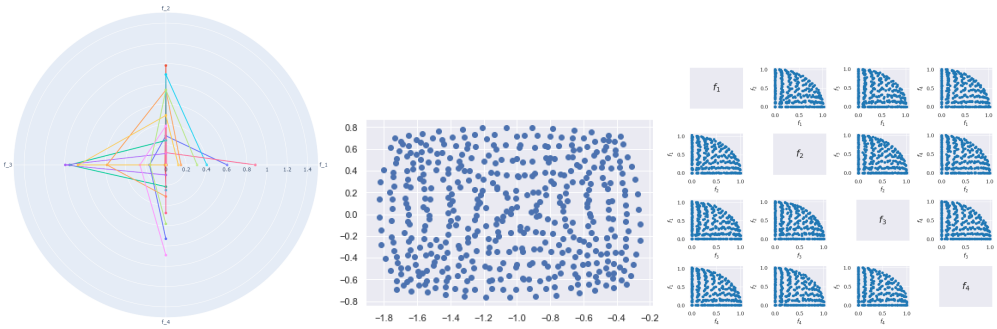
Whilst scarce, attempts have been made to visualise the evolutionary process. For example, the work visualising the population entirety proposed by Walker, Fieldsend & Everson (2012) for which they demonstrate a number of techniques. One is the seriation of heatmaps which we discussed earlier. Another method is Pareto shells which use the methods of Pareto sorting from NSGA-II to create a partial ordering which can be visualised with a graph representation. The Pareto sorting orders the solutions into shells based on their domination hierarchy. The solutions are represented as individual nodes with the solution index number inside each node. Solutions are plotted in columns. Each column represents a shell and directed edges from shell i to $i+1$ points from the solution dominating to the solution dominated. The work also utilises a second metric for colouring each solution based on average rank. Average rank considers ranking the objective of each solution against all other solution objectives and then taking an average of all objective ranks for each solution. The solution can then be coloured based on this average rank value and give an indication of whether a solution is a strong all-around solution or whether it just dominates on a single objective and is poor on all other objectives. A limitation of this work includes difficulties visualising a large number of solutions or high-dimensional space where the probability of being non-dominated is exponentially increasing.



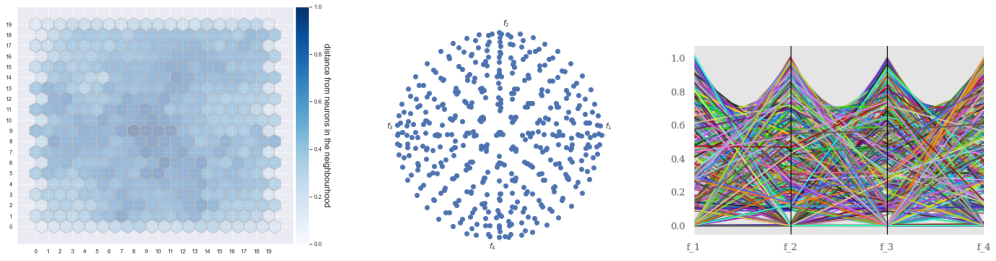
(a) 3-D RadViz *(b)* GTM *(c)* Heatmap



(d) MDS *(e)* PCA *(f)* Prosection Method



(g) Radar *(h)* Sammon Method *(i)* Scatter matrix



(j) SOM *(k)* Star Coordinate *(l)* PCP

Figure 2.6: Visualisations of common Pareto front display methods.

Isomap (Tenenbaum et al. 2000) was shown to visualise EA solutions (Walker et al. 2010) and populations (Walker, Fieldsend & Everson 2012). Isomap aims to reduce the dimensions of the population by preserving the geodesic distance; that is, it seeks to preserve the pairwise distances between solutions. To transform the population solutions using Isomap, we compute a graph of nearest neighbours for each solution where each edge represents the euclidean distance between each solution. We can then project the solutions into a lower manifold space whilst maintaining these pairwise distances.

Other dimensionality reduction techniques such as multi-dimensional scaling (MDS), have been used to visualise populations (Walker, Everson & Fieldsend 2012). The purpose of MDS for population dimensionality reduction is to derive a set of vectors which reduces the EA population to a lower dimensional embedding whilst preserving pairwise population distances. Formally: given a pairwise $N \times N$ distance dissimilarity matrix $\mathbf{D} = (d_{i,j})$, find N vectors $x_1, \ldots, x_N \in \mathbb{R}^K$ such that $||x_i - x_j|| \approx d_{i,j}$ for all $i, j \in \{1, \ldots, N\}$. The notation $|| \cdot ||$ denotes the standard Euclidean norm. We discuss MDS in further detail in Chapter 3.

The work of Walker (2015) proposed the use of treemaps (Johnson & Shneiderman 1998) to visualise a multi-objective population in terms of solution dominance. A node is represented for each dominating solution, with edges linking the solution to other solutions. It dominates until we have a tree of solutions, with the top representing dominating solutions and the leaves representing the most dominated solutions. Therefore, the dominating solution size area is equal to the total of its subcategories (solutions that it dominates). This tree can be used to create a visualisation known as a treemap in which polygons, usually squares and rectangles, are used to represent solutions with the area of the polygon indicating the solution value and dominated solutions falling within the area of the dominating solution. Treemaps can be effective at determining clusters and like Pareto shells are good for showing dominance hierarchy relationships.

The effectiveness of visualisations to intuitively enhance and complement algorithms has not gone unnoticed. The work by Walker & Craven (2018*b*) visualises the population for examining algorithm parameters by mapping individuals from the high-dimensional objective space to a 2-D polar coordinate space and representing whole populations as a single actor. The actor moves around the polar space as it converges, measured with hypervolume, on the Pareto front. The diversity is measured with the crowding distance and indicated in the visualisation with colour. Other works visualising the performance of EAs include Walker & Craven (2018*a*) - these consider visualising performance indicators of EAs with different parameters. Other studies illustrating algorithm performance are that of Craven & Jimbo (2014) which considers a single-objective cryptography problem.

Later work such as De Lorenzo et al. (2019) compared the extent to which dimension reduction techniques preserved population movements and the exploration-exploitation trade-off using single-objective problems. It also proposed two compact visualisations. We detail this work in Chapter 3.

A popular population quality diversity algorithm includes the work of Mouret & Clune (2015), which considers the use of solution filters to reduce a large search space to a

more manageable space using the structures within sets of solutions. Furthermore, the diversity usage (DU) maps (Medvet et al. 2018) provides a heatmap visualisation of a population of genotypes showing how each gene contributes to the solution (diversity and usage of the gene).

2.4.5 Visualising Operators

This subsection considers the existing visualisation literature that concerns itself with providing visualisation which directly or indirectly shows the effect operators have on the search. Visualisations which display the effects of operators can create a more informative picture of the optimiser's search and also be used to analyse the effect of the operator on generating solutions. The literature for the visualisation of operators is very limited in the EA domain, although far more prevalent in the GP domain (Daida et al. 2004, Burlacu et al. 2013, McPhee et al. 2016, Sekanina & Kapusta 2016, McPhee et al. 2017, 2018).

With regards to existing literature that can visualise operators for an EA search, the work of Črepinšek et al. (2011), proposed a novel tree structure to examine exploration and exploitation in EAs; the natural ability of graphical trees allows one to examine the lineage between solutions (i.e., visualise which parents are related to which solutions). The work also displays the operator used for reproduction. The work considered applying the method to a multi-objective 0/1 knapsack problem.

Furthermore, pedigree graphs or family trees predicated upon genealogy can also visualise parent-offspring relationships. However, one cannot determine the operator used for reproduction during the search. Furthermore, the visualisation does not display the solutions in the objective space, and so considerable information is lost regarding the search. Other work (Keshk 2017) attempts to alleviate this issue by visualising pedigree graphs in conjunction with a fitness graph.

The work of Wu et al. (1999) attempts to implement a barcode to display information in a visual way. The barcode is a representation of the genetic material within the solution. From this visualisation, it is possible to determine the operator and parental solutions used to generate the solution. However, whilst informative, the visualisation of large barcodes can cause an abundance of information and make comparing and interpreting solutions challenging.

2.4.6 Landscape Visualisation

The visualisation of landscapes is essential when considering landscape analysis to gauge a deeper understanding of the problem landscape. A large share of this work is centred around LONs (Ochoa et al. 2008). This work uses ideas from the energy landscape of atomic cluster analysis and the field of complex networks. It introduces analytical techniques to the study of fitness landscapes initially for combinatorial optimisation. The work utilises a local search to find all optima and basis of attraction, then considers a graph theory representation of combinatorial fitness landscapes, where such fitness analysis could benefit heuristic design/evaluation. The graphic theory representation uses nodes S^* to represent optima and edges E to represent basins of attraction; this representation is known as a local optima network (LON). Formally, a LON can be represented as the graph $G = (S^*, E)$. The study uses combinatorial NK-

landscapes as the evaluation landscapes. For NK-landscapes, there exist N binary variables in each genotype, and K controls the number of genes that influence a particular gene. Furthermore, the work provides an analysis of these landscapes to reveal and confirm interesting landscape characteristics such as the optima distribution is not random, optima distribution is higher around the global optima, and optima degree distribution is closer to exponential than Poissonian as initially expected. A significant body of work was produced from the initial work on LONs. Later work considered LONs with neutrality (neighbouring solutions can yield the same fitness) (Verel et al. 2010) and LONs with escape edges (edges with a distance condition) (Verel et al. 2011). LON analysis with six other methods (Chicano et al. 2012) and LONs for continuous fitness space (Adair et al. 2019). Visualising these graph representations by plotting the nodes and edges Ochoa et al. (2014).

The LONs work has been adapted to compare and evaluate metaheuristic performance with a data-driven approach, in which they propose Search Trajectory Networks (STNs) (Ochoa et al. 2020) as an alternative to data reduction. The STN model is a graph object like a LON. LONs, however, model the fitness landscape, whereas STN models the search behaviour of a metaheuristic during optimisation. The nodes N, therefore, are locations in the search of sampled solutions; the edges E are connections to two consecutive locations, which are weighted based on the frequency of transition. Later work (Ochoa et al. 2021) provided STN methodology for both population-based heuristics (as in the original work) and single point local heuristics in both combinatorial and continuous space. In Lavinas et al. (2022), the STN work was extended to multi-objective algorithms with the use of problem decomposition. Like LONs, the graph objects can be visualised with the support of force-directional layout algorithms to optimise interpretability and illustrations are provided in the work. The visualisation provides information on the strong search areas/most attractive and the solution convergence and exploration to provide insight into the metaheuristic behaviour.

The work of Schäpermeier et al. (2020) considers visualising the decision space of continuous multi-objective problems (MOPs) proposing a 'PLOT' visualisation comprised of a gradient field heatmap and cost landscapes. Whilst this method does not visualise the entire population, one can observe some problem features such as local optima, basins of attraction and global optima. It notes visualisation techniques for the decision space of continuous MOPs are scarce in research (there being only two existing visualisations that consider the decision space) (Schäpermeier et al. 2020).

2.4.7 Interactive Visualisation

The benefit of interactivity for solution visualisation is a tailored approach to better suit the DM and the problem. This allows one to potentially remove superfluous information or focus on specific solutions or aspects of the search eliminating some of the issues with conventional visualisations which may overwhelm the DM. In contrast, interactive visualisation requires significant time to develop the application and may have hardware requirements which do not come with conventional non-interactive visualisations.

The work of Varga et al. (2021) proposed an interactive visualisation by reducing the objective values for many-objective solutions to two objectives with PCA. They utilise geometric ions (geons) to represent solutions and their original objective values. The

work capitalises on recognition by components theory (Biederman 1987) to use the geons characteristics like symmetry, size and colour to represent the objective values. The geon has a large main body to represent its positions in the dimension reduced space, then smaller objects are attached to the body to indicate the objective number, the size of that attached object indicates its objective value, thus for a 3-objective solution, one would expect to see three objects attached to the main body of the geon. This is one of the few works which propose a usability study of the visualisation in the literature to support the results; however, they note a number of limitations for the usability study.

The interactive tools ELICIT (Cruz et al. 2015) and GAVEL (Hart & Ross 2001) are both visualisation tools that can display lineage among solutions and exhibit the operators used to reproduce solutions. Whilst providing a wealth of information, the objective space population structures cannot be identified in the visualisation. ELICIT is designed to provide visual exploration of the evolutionary algorithms presenting tree and graph based models to visualise the decision variables and the objective variables. For GAVEL, the visualisation shows the best solution for the EA run. The horizontal axis indicates the solutions' generations. The left column displays the generation number with the chromosome displayed in the window. The chromosome is connected to their parents by a line to create an ancestry tree.

The EAVis tool (Kerren & Egger 2005) can visualise operators facilitating solution perturbation during optimisation; this information is conveyed in a table-like format in conjunction with separate fitness plots. The GeneaQuilts (Bezerianos et al. 2010) tool provides a compact visualisation displaying EA genealogies. One of the most recent tools is VisEvol (Chatzimparmpas et al. 2021); this provides visual analytics tools for optimisation. The purpose of this tool is to support the optimisation of machine learning hyper-parameters with an evolutionary algorithm. This tool can show a variety of existing visualisations, many of which may together convey a plethora of information. One can also use animations of scatter plots to evaluate MOEA algorithm performance (Chakuma & Helbig 2018). The animation feature allows one to add and consider the dimension of trajectory to the visualisation.

Due to the scope of this work being visualisation, we do not discuss at depth literature considering other mediums of communication but do recognise that work for other mediums, such as using audio cues to facilitate EA insight, does exist. For example, the work of Asonitis et al. (2022) proposes SonOpt, which is a sonification tool for monitoring algorithm convergence, shape and diversity, supporting greater accessibility to visually impaired users as well as an alternative to visualisation (perhaps at the loss of some clarity and a reduced number of communication mediums).

2.5 Summary

This chapter has described the existing background material required to motivate future technical chapters. The chapter considers the EA process, AI explainability methods, the nascent explainability for EC methods and the existing visualisation work within EAs. In the upcoming chapters, we will alleviate some significant issues associated with the current existing literature.

Chapter 3

Visualising Evolution History in Multi- and Many-Objective Optimisation

The chapter extends an existing parameter space visualisation method to visualise the objective space and then visualise multi- and many-objective problems. Furthermore, we adapt the method for arbitrary runtime use to mitigate the computation complexity limitations of previous methods. For the problems tested in this chapter, the proposed method can create visualisations that are visually akin to the previous parameter space method whilst requiring less than 1% of the time and memory necessary to visualise the same objective space solutions. Finally, quantitative and qualitative analysis is performed on two proposed novel methods using solutions from multi- and many-objective problems, revealing the final proposed visualisation method to be a potentially informative tool for understanding population dynamics with low computational complexity. The material in this chapter has been published as two peer-reviewed outputs:

- *Walter, M. J., Walker, D. J. & Craven, M. J. (2020), Visualising evolution history in multi- and many-objective optimisation, in 'International Conference on Parallel Problem Solving from Nature', Springer, pp. 299–312.*
- *Walter, M. J., Walker, D. J. & Craven, M. J. (2022b), Visualizing population dynamics to examine algorithm performance, IEEE Transactions on Evolutionary Computation 26(6), 1501–1510.*

EVOLUTIONARY algorithms (EAs) evolve solutions utilising numerous reproduction operators and in a stochastic nature which creates difficulties in comprehending the dynamics of the population to gauge a better understanding of how the solutions are evolving through a search space. Population dynamics (also referred to as population trajectories) are important to support the understanding of landscape analysis, algorithm enhancement, algorithm development and explainability. Ultimately, comprehension of an EA evolving through a search space would benefit any user and application of EAs.

Due to the human ability to quickly detect visual patterns, visualisation of the EA process is a natural approach. Notwithstanding, challenges emerge when endeavouring to visualise low-dimensional solutions and compacting the extensive amount of information yielded by EAs (large population sizes and generations) so that it is manageable for humans to comprehend, usually at the cost of accuracy. One way to attempt to understand population dynamics is by visualising the population of an EA as it evolves. For example, one could use a scatter plot to visualise single-objective or dual-objective populations. However, many problems are multi- or many-objective. A study of the problem properties for industrial and academic real-world EA optimisation problems (van der Blom et al. 2020) showed 33% of problems were multi- or many-objective

(based on a sample population size of 45 participants). This number could be even larger with greater understanding, development and use of EAs for real-world optimisation.

As reviewed in Section 2.4.4, some work exists to visualise multi- and many-objective populations in two dimensions with dimension reduction techniques (e.g., using PCA to visualise the first two principal components). Visualisation in three dimensions relies on visualising the top three principal components. Using well-defined mathematical methods such as dimension reduction techniques is often supported by rigorous mathematical proofs, such as using the manifold hypothesis that states real-world low-dimensional data lie in a lower-dimensional embedding (Fefferman et al. 2016). The cited work allows one to gauge the population from a spatial perspective but does not allow one to understand the solution dynamics; rather just informs one where the solutions have been during the search process. One could colour the population based on the generation it resides in to attempt to understand the population dynamics; however, the population trajectories can still be unclear, particularly if solutions are revisiting or remaining in previously explored areas of the search space. When solutions are perturbed to the same area of the search space as previous generations, the new solutions will overwrite existing solutions (a many-to-one mapping). To truly begin to comprehend solution dynamics, one needs to consider time. Moreover, one needs to construct a visualisation which can exhibit the change in solutions over time.

This chapter is structured as follows. Section 3.1 considers a single objective visualisation of the parameter space search which we later adapt. This includes the methodology from the work of De Lorenzo et al. (2019), and we provide the reasons for choosing this method to adapt, along with its limitations. Section 3.2 details the adaption of the existing methodology to multi-objective problems with a particular focus on the extension of visualising the objective space rather than the parameter space. The results and the experimental setup, containing details of the parameters used, are highlighted in their respective sections. The many-objective examples are highlighted in Section 3.3, and the strengths and limitations of this method are considered, which motivates a further adaption to reduce the complexity of the methods for use in benchmark problems. Section 3.4 and Section 3.5 propose and detail the methodology of the population dynamics visualisation methods, which have been adapted from the previous works. A comprehensive analysis of results to evaluate the performance of the adaptation using both qualitatively (graphical interpretation) and quantitative (statistical ranked root mean square error (RMSE) evaluation) representation methods are performed in Section 3.6. We analyse how the visualisations can offer far greater insight into algorithm performance than using traditional algorithm performance metrics such as hypervolume alone, and can be used to complement explicit performance metrics. Finally, a conclusion is provided for this chapter in Section 3.7.

3.1 Single-Objective Visualisation of the Parameter Space Search

The work of De Lorenzo et al. (2019) considers the variable of time in a population visualisation. Allowing a visual reference to time supports the understanding of population dynamics as one can understand the solution movements with respect to time. The cited work considers four different dimensionality techniques (PCA, MDS, UMAP

(McInnes et al. 2018) and t-SNE (Van der Maaten & Hinton 2008)) to create two different visualisations (a series of 2-D scatter plot frames per generation and a 3-D scatter plot with time as an axis). They also assess the ability of these dimension reduction techniques to preserve the population dynamics by measuring the exploration-exploitation trade-off and solution movement. The work considers single objective continuous and binary problems and attempts to visualise the parameter space. The population size used for experimentation is 50, and the number of generations is 50, totalling 2500 function evaluations with the parameter lengths varying in size - noting quite a small search.

The cited work only considered visualising the parameter space. The objective space would reveal the solutions' fitness, which is usually the primary consideration of solution selection by a DM (Decision Maker). However, the parameter space allows a DM to gauge some insight into how the parameters affect the objective values and support the localisation of robust solutions, i.e., from an objective space visualisation, two sets of solutions may yield strong fitness. However, to choose between the two, it could be the case that a small perturbation in the parameter space results in a more substantial decline in fitness for one of the sets. So we argue that both spaces contain essential information about the search, although, of the two, the objective space is the primary determinant of a good solution.

3.1.1 Visualising Search Parameter History

The visualisation method used herein is based on that defined by De Lorenzo et al. (2019); we currently do not adapt the method for visualising the search parameter space to recreate and evaluate the method and results, which are later compared with the methods of this thesis. The method attempts to visualise an optimiser's search history, specifically the parameter values in their respective generation, once the optimisation process has been completed. The optimiser results in a sequence of populations in which P_i is the population from the i-th generation ranked according to its members' fitness values, where $i \in \{1, \ldots, n_{gen}\}$, and n_{gen} is the total number of generations. This sequence of populations is concatenated into a single multiset with the number of columns representing the number of parameters, D, plus an additional column to contain the generation value, gen. The first D column's dimensionality is reduced using some dimensionality reduction technique from D to 2. The mapping Π is a dimension reduction mapping if $\Pi : \mathbb{R}^D \to \mathbb{R}^n$, where $n < D$. The dimensionality technique is used to translate pairwise distances of the population individuals into a lower-dimension Cartesian space which can now be visualised in two dimensions. For values of $D < 3$, dimensionality reduction would not be required. For all examples in this work, $D > 2$.

Once the dimensionality of the solutions is reduced, the resulting embedding is then used for visualisation; in the original work, two visualisations are produced. The 2-D scatter plot visualisation considers plotting the two embedded coordinates in the x and y coordinates with $z = 0$ for all solutions. There is n_{gen} number of plots. Therefore for a 10 generation run, one would be required to plot ten 2-D scatter plots to visualise every generation. On the other hand, the 3-D scatter plot visualisation is of greater interest to this work because of its ability to visualise the element of time (generations) in a single compact plot. In this case, only a single plot is required to visualise all generations of

any value of n_{gen}. The 3-D scatter plot visualisation is produced by the two embedded coordinates forming the x and y coordinates, and the generation number gen provides the value for the z-axis. Within the visualisation, colour is used to illustrate the trade-off between search and exploitation, showing in which mode of optimisation the algorithm is currently operating. The works of Črepinšek et al. (2013) and De Lorenzo et al. (2019) are employed to determine to what level the set of all solutions at a particular generation is being explored or exploited. Exploration is inversely proportional to exploitation. This metric is applied to the visualisation in De Lorenzo et al. (2019). See the next subsection for more detail on the methodology.

3.1.2 The Exploration Exploitation Metric

Visualisations are coloured depending on the continuous state in which the algorithm is operating and to which degree the algorithm is exploring or exploiting the search space. This state is a continuous metric yielding a value in the $[0, 1]$ range. The exploration-exploitation trade-off metric is used in De Lorenzo et al. (2019), but proposed by Črepinšek et al. (2013). In the latter work, they note that the processes of exploration and exploitation have not previously been formally defined but rather given naive definitions. The exploration and exploitation metric can be defined in several ways. This includes measuring the similarity of an individual with the parent(s), the similarity with the most similar individual in the whole population during an evolutionary run, the similarity to the most similar individual in a subset of the population (such as the same generation as the individual) or similarity to the most similar individual in the whole population.

The work of De Lorenzo et al. (2019) measured the similarity of each solution to its most similar solution in the previous generation. This thesis defines this in equation 2.7. The similarity was defined as the Euclidean distance for continuous problem solutions and the Hamming distance for binary problem solutions. The Hamming distance can be defined as, given two strings of equal length, the minimum number of character substitutions required to make the strings equal. The work used this metric to calculate the exploration and exploitation status in both the original low-dimensional space and for the solutions in the low-dimensional embedding. Then, a comparison is made to gauge how much exploration and exploitation information is lost in the dimensionality reduction process for each method.

The exploration-exploitation trade-off metric is calculated for each solution in each generation g. We first calculate the Euclidean distance to the solution's closest neighbour (the Euclidean distance between each pairwise individual in the population is calculated earlier) and store these values in the similarity to the closest neighbour (SCN) array. This is the nearest neighbouring solution in the population up to the generation of the solution being measured. The median, SCN^*, of SCN is calculated. For all $SCN_g > SCN^*$ we add 1 to the exploration-exploitation trade-off metric, τ_g, for each generation. Once all τ_g values have been calculated for every generation, we normalise the values of τ_g, so the solutions can be scaled and coloured according to the τ_g value of each solution. Individuals with a lower τ_g or distance between the most similar neighbours are considered to be exploiting.

3.1.3 Dimensional Reduction Methods

Dimensionality reduction is commonly used for feature analysis, filtering noise and reducing data complexity. One purpose for reducing the complexity of the data is to make low-dimensional data visualisable without losing too much information from the mapping between spaces. What separates the different dimensionality types is the method of mapping. The dimensionality method is chosen based on the type of data (i.e., data with linear trends or qualitative data) and the information to be preserved (i.e., Euclidean distance or variance). In this Section, we detail the four dimensionality reduction techniques, namely: PCA, MDS, UMAP and t-SNE, evaluated in De Lorenzo et al. (2019). In addition, see Section 2.4.3 for an overview of the existing dimensionality reduction with respect to the EA visualisation literature.

One dimension reduction method used in the work of De Lorenzo et al. (2019) is principal component analysis (PCA). The purpose of PCA is to derive a set of vectors which reduces the dataset (EA population) to a lower-dimensional embedding whilst preserving the covariance of the data. The aim is to obtain a sequence of unit vectors. Each unit vector is orthogonal to its previous unit vector and is the direction of the line of best fit for the data. The output results in the unit vector (eigenvector) and eigenvalues. The eigenvectors of the data's covariance matrix are the principal components. The eigenvalues provide the variance captured by each eigenvector respectively and can be visualised on a scree plot. The m eigenvectors with the greatest eigenvalues are chosen, and the rest are discarded. Commonly, singular value decomposition (SVD) is used to compute the eigendecomposition of the data's covariance matrix.

SVD is used to compute a lower rank (fewer columns/dimensions) matrix, which approximates the original matrix. The idea is that any matrix, x, can be decomposed into a product of three matrices:

$$X = U\Sigma V^T \tag{3.1}$$

These three matrices can be considered linear transformations of the data, such as V being a rotation matrix. The covariance of the data matrix X is $X^T X$, and thus:

$$X^T X = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T \Sigma V^T = V\Sigma^2 V^T \tag{3.2}$$

The matrices U and U^T can be reduced since $U^T U$ is the identity matrix, and Σ is a square diagonal matrix.

MDS aims to derive a set of vectors that reduces the dimensionality of a dataset, such as a low-dimensional EA population, to a lower dimensional embedding whilst preserving pairwise population distances. More formally: given a pairwise $N \times N$ distance dissimilarity matrix $\mathbf{D} = (d_{i,j})$, find N vectors $x_1, \ldots, x_N \in \mathbb{R}^K$ such that $||x_i - x_j|| \approx d_{i,j}$ for all $i, j \in \{1, \ldots, N\}$. The notation $||\cdot||$ denotes the vector norm, which for the case of this work is the standard Euclidean norm.

The three frequently adopted MDS algorithms are classical MDS (also known as Principle Coordinates Analysis (PCoA)), metric MDS and non-metric MDS. All three MDS algorithms require a similarity matrix input, though the algorithm employed depends

on the input matrix. For example, for non-metric MDS the similarity matrix is qualitative (given by rank only). The work of De Lorenzo et al. (2019) does not specify the type of MDS utilised. However, we produced very similar visualisations after computing the solution coordinates with both classical MDS and metric MDS. For the work in this chapter, we consider classical MDS.

Metric MDS works by minimising a stress function, i.e., the residual sum of squares. Thus, given a pairwise dissimilarity matrix $\mathbf{D}$ with elements $d_{i,j}$, we aim to find $x_1,\ldots,x_N \in \mathbb{R}^k$ (where k is the reduced dimension - which is two for these visualisations) which minimise pairwise distance errors:

$$Stress_D(x_1,x_2,\ldots,x_N) = \left(\frac{\sum_{i,j}\left(d_{i,j}-\left\|x_i-x_j\right\|\right)^2}{\sum_{i,j} d_{i,j}^2}\right)^{\frac{1}{2}} \tag{3.3}$$

Classical MDS employs eigenvalue decomposition (utilising SVD) to find a lower dimension k embedding matrix $\mathbf{Y}$, which approximates the original matrix. Initially, we create the $N \times N$ squared proximity matrix $\mathbf{D}^{(2)} = \left(d_{i,j}^2\right)$. Utilising the centring matrix $\mathbf{C} = \mathbf{I} - \frac{1}{N}\mathbb{O}$ (where $\mathbf{I}$ is the identity matrix and $\mathbb{O}$ is the $N \times N$ matrix filled with ones) double centring is then applied to $\mathbf{D}^{(2)}$ to create matrix $\mathbf{B}$:

$$\mathbf{B} = -\frac{1}{2}\mathbf{C}\mathbf{D}^{(\mathbf{2})}\mathbf{C} \tag{3.4}$$

With $\mathbf{B} = \mathbf{Y}\mathbf{Y}'$, since $\mathbf{B}$ is symmetric (and now centred), we can use eigenvalue decomposition to decompose $\mathbf{B}$ into $\mathbf{E}\Lambda\mathbf{E}'$, where $\mathbf{E}$ is a matrix of eigenvectors and Λ is a diagonal matrix of eigenvalues to determine the k largest eigenvalues $\lambda_1,\ldots,\lambda_k$ and their corresponding eigenvectors (non-positive eigenvalues are ignored). Finally, the $k \times N$ matrix $\mathbf{Y}$ can be constructed by

$$\mathbf{Y} = \mathbf{E}_k \Lambda_k^{\frac{1}{2}}. \tag{3.5}$$

Both PCA and MDS preserve the linear structure of the data, but it could be the case that non-linear relationships exist among the data, so preserving these non-linear relationships is essential for preserving maximum information upon the reduction of data dimensionality for non-linear data. Furthermore, PCA is only effective if most of the data variance is in two dimensions, making it not suitable for more complex data. To consider the non-linear relationships, the authors of De Lorenzo et al. (2019) experiment with two non-linear preserving methods: t-SNE and UMAP. These methods, like PCA and MDS, are commonly found in the machine learning literature. The t-distributed stochastic neighbour embedding (t-SNE) (Van der Maaten & Hinton 2008) seeks to map high-dimensional objects to a lower-dimensional space aiming to group similar objects as neighbours with a high probability when projected into the lower-dimensional space. When objects are dissimilar in the high-dimensional space, the probability of them neighbouring in the lower-dimensional space is lower. Whilst PCA aims to preserve variance (dissimilarity), t-SNE aims to preserve similarity between points. This often results in a clustering of points that are similar in the high dimensions, which is dependent upon the parameters, thus sometimes leading to false clusters (patterns

that do not exist in the low-dimensional data). For large datasets, the technique can be implemented using a faster ($O(N\log N)$) Barnes-Hut approximation (Barnes & Hut 1986).

One parameter is perplexity. Perplexity can be considered the number of nearest neighbours whose best value depends on the density of the dataset. So the larger the density, the larger the perplexity value should be. However, the perplexity value should avoid being so large that all points are near equidistant. Perplexity values usually range from 5 to 50 (Van der Maaten & Hinton 2008).

The t-SNE algorithm works by first determining the similarity of all the points in the high-dimensional space. This is done by measuring the distance between two points in the high-dimensional space. The distance between the two points is considered on a Gaussian normal curve for which the mean is the point of interest. For larger distances between a point j and the point of interest i, the normal value p_{ij} will be lower (further along the normal curve). This is done for all points. So, for all points, there is a matrix of similarity values to all other points. These similarity values need to be then scaled to add up to one - this is so less dense clusters have wider normal curves (greater standard deviation). Thus all clusters are on the same similarity scale regardless of the density of the cluster.

For $i \neq j$, define

$$p_{j|i} = \frac{\exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2 / 2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-\left\|\mathbf{x}_i - \mathbf{x}_k\right\|^2 / 2\sigma_i^2\right)}$$

and set $p_{i|i} = 0$. Note that $\sum_j p_{j|i} = 1$ for all i.

Because the width of the distribution is based on the surrounding data points' density, the similarity score between two points can be different depending on the point of interest (i.e, $p_{ij} \neq p_{ji}$), t-SNE averages 'symmetrises' the similarity scores between two points, so they both yield the same value. The probability that datapoint i and j are similar is denoted p_{ij} where

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}, \tag{3.6}$$

and note that $p_{ij} = p_{ji}, p_{ii} = 0$, and $\sum_{i,j} p_{ij} = 1$.

We then want to plot the points in the lower dimensional embedding maintaining the similarity score from the low-dimensional space. First, we randomly project the points in a lower-dimensional embedding, and then we calculate the similarity score between points as done previously in the higher dimensional space; however, this time with a t-distribution rather than a normal distribution. This is because the longer tails of the t-distribution are thought to stop the clusters from clustering too densely to visualise any useful information. Using a student t-distribution with longer tails than the Gaussian distribution means points that are close together in the low-dimensional space are closer (compressed together) in low-dimensional embedding, and solutions far apart in the low-dimensional embedding are pushed further apart in the low-dimensional embedding. The similarity score calculated with the t-distribution with a single degree of freedom can be computed by:

$$q_{ij} = \frac{\left(1 + \left\|\mathbf{y}_i - \mathbf{y}_j\right\|^2\right)^{-1}}{\sum_k \sum_{l \neq k} \left(1 + \left\|\mathbf{y}_k - \mathbf{y}_l\right\|^2\right)^{-1}} \tag{3.7}$$

Then, t-SNE moves each point in the lower dimensional space to minimise the scaled similarity value between the high and low-dimensional points. This uses a gradient descent method based on Kullback-Leibler divergence:

$$\mathrm{KL}(P\|Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{3.8}$$

Unfortunately, the t-SNE algorithm scales poorly with high memory complexity, especially for large perplexity values because of the k-nearest neighbour step. Furthermore, t-SNE preserves local structures but does not preserve the global structure of the data, meaning solutions within the same cluster are similar; however, clusters that are next to other clusters are not necessarily similar. The t-SNE algorithm requires the low-dimensional input to be reduced with a dimensionality reduction technique before inputting into t-SNE. An alternative but similar algorithm is the UMAP algorithm, and this is the final dimension reduction technique used in the work of De Lorenzo et al. (2019).

UMAP is similar to t-SNE; however, it differs in a few ways, which we outline below. UMAP works by attaining similarity scores in the low-dimensional space. Similarity scores for each data point use an exponential probability distribution. t-SNE uses the perplexity parameter to alter the shape (width and height of the curve) of the Gaussian distribution (the y values of the Gaussian distribution are the similarity scores); however, whilst similar, in UMAP, we explicitly set the number of low-dimensional neighbours k for each point. The formula is:

$$p_{i|j} = e^{-\frac{d\left(x_i, x_j\right) - \rho_i}{\sigma_i}} \tag{3.9}$$

As we use different distributions for each point in UMAP and t-SNE, we get asymmetrical distances between points, that is, the weight of the graph between points a and b are same as the weight from point b to point a. t-SNE averages the similarity score between two points. However, UMAP uses a slightly different way to compute symmetry, as defined by:

$$p_{ij} = p_{i|j} + p_{j|i} - p_{i|j} p_{j|i} \tag{3.10}$$

UMAP then initialises the solutions in the low-dimensional space. Whilst t-SNE uses the Student t-distribution for probabilities in the low-dimensional embedding, UMAP uses different curves. To calculate low dimensionality similarity scores, UMAP uses the formulae:

$$q_{ij} = \left(1 + a\left(y_i - y_j\right)^{2b}\right)^{-1} \tag{3.11}$$

Where d is the low dimensional distance between two points. Whilst a and b control how packed together the points are; giving one more control (than t-SNE) over how tightly packed the low-dimensional data is plotted. Then UMAP uses binary cross-entropy (CE) rather than KL-divergence, allowing UMAP to capture the global structures too. Stochastic gradient descent is used to minimise at a low computation complexity cost. Furthermore, in the high or low dimensional space, normalisation does not occur. This also reduces computing time.

$$CE(X,Y) = \sum_i \sum_j \left[p_{ij}(X) \log \left(\frac{p_{ij}(X)}{q_{ij}(Y)} \right) + (1 - p_{ij}(X)) \log \left(\frac{1 - p_{ij}(X)}{1 - q_{ij}(Y)} \right) \right] \tag{3.12}$$

3.1.4 Analysis of the Parameter Visualisation Method

We start by recreating a subsection of the existing work of De Lorenzo et al. (2019) to motivate and understand some of the challenges we later address. The work of De Lorenzo et al. (2019) considers the application of the previously discussed dimension reduction methods on solutions generated by an optimiser on a simple bit string binary optimisation problem and a continuous optimisation problem. We formulate these problems in the paragraph below.

The fitness is defined as minimising the distance from the solution to the closest optimum $x_i^\star$, more formally: $f(x) = \min_{x^\star \in X^\star} d(x, x^\star)$. For the cited work's binary problems, each solution x is a bit string $x \in \{0,1\}^l$, where l is the length of the string. There are n optima x^*. For the examples in this chapter and the cited work, $n = 1$ and the distance metric implemented is the Hamming distance. The continuous problem solutions $x \in \mathbf{R}^l$ are also optimised to minimise the Euclidean distance from the solution to the closest optimum in the predefined n optima set $\mathbf{X}^\star = \{\mathbf{X}_1^\star, \ldots, \mathbf{X}_n^\star\}$. For examples in this section the optimum is defined $\mathbf{x}_i^\star = \left(x_{i,1}^\star, \ldots, x_{i,l}^\star\right)$ for which:

$$x_{i,j}^\star = \begin{cases} \cos\left(i\frac{2\pi}{n}\right)\left(\sin\left(i\frac{2\pi}{n}\right)\right)^{j-1} & \text{if } j \neq l \\ \left(\sin\left(i\frac{2\pi}{n}\right)\right)^{j-1} & \text{otherwise} \end{cases} \tag{3.13}$$

The work of De Lorenzo et al. (2019) initially considers the quantitive statistical comparison of the solutions in the original objective space with the lower-dimensional embedding space solution representation. The paper assesses the correlation between the exploration-exploitation metric τ in the reduced and original objective space; a similar result can be seen in Figure 3.1. We observe a strong positive correlation between the MDS-reduced solutions and the initial search space solution, as seen from the graph. However, t-SNE has no clear positive correlation. Furthermore, whilst MDS reflects the same exploration-exploitation trade-off as represented in the search space, for which exploration is initially high before reducing over time, t-SNE shows no decline in exploration.

A measure of the preservation of population movement is evaluated by considering the trajectory (movement between generations) of the best individual solution in the original search space and the lower-embedding space to obtain the inter-generation distances. The results we obtained by recreating the work supports the results of the original

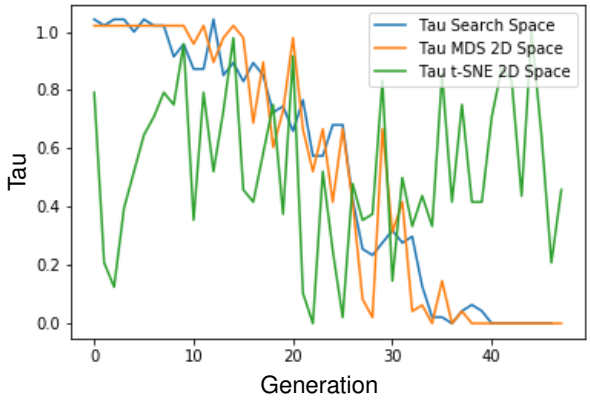


Figure 3.1: Exploration rate trade-off in the search space, MDS space and t-SNE space for a continuous optimisation problem. The parameters: $l = 20$, $n = 1$.

Mean (μ) and Standard Deviation (μ)				
Method	$l = 2$	$l = 5$	$l = 10$	$l = 15$
PCA	1.00 ± 0.00	0.97 ± 0.03	0.94 ± 0.06	0.93 ± 0.03
MDS	0.96 ± 0.03	0.95 ± 0.02	0.94 ± 0.03	0.93 ± 0.06
t-SNE	0.18 ± 0.13	0.21 ± 0.26	0.02 ± 0.22	-0.04 ± 0.26
UMAP	0.17 ± 0.13	0.24 ± 0.25	0.20 ± 0.20	0.05 ± 0.18

Table 3.1: Pearson correlation means and standard deviations from 10 repetitions. The closer to 1, the more inter-generational distance correlation between the best solution in the original and lower-embedding. Parameters are set to $n = 1$ and $l = 5$.

paper; for example, MDS shows a good correlation of 0.96, which indicates MDS's strong ability to produce good representations of the corresponding best individual's distance in the search space — followed by PCA, t-SNE and UMAP. Moreover, by observing the standard deviation of the correlation across the independent runs for each combination, it can be seen that the variability of this index is lower for MDS and PCA than for t-SNE and UMAP.

Furthermore, from the results we obtained, we calculated the Pearson coefficients for the best individual inter-generational distance for ten runs; this can be found in Table 3.1. With this data, we can concur that MDS and PCA show a much greater correlation than t-SNE and UMAP, as well as MDS and PCA yielding a smaller standard deviation than that of t-SNE and UMAP. Pearson's correlation was calculated for all four dimensionality reduction techniques, and we can rank them based on the exploration-exploitation metric in the following order (from best to worst) PCA, MDS, t-SNE and UMAP. A similar analysis is done for the binary problems which yield similar results to continuous problems. The paper then considers a qualitative analysis of the visualisations.

To summarise the findings of the work, PCA and MDS provide the highest Pearson's

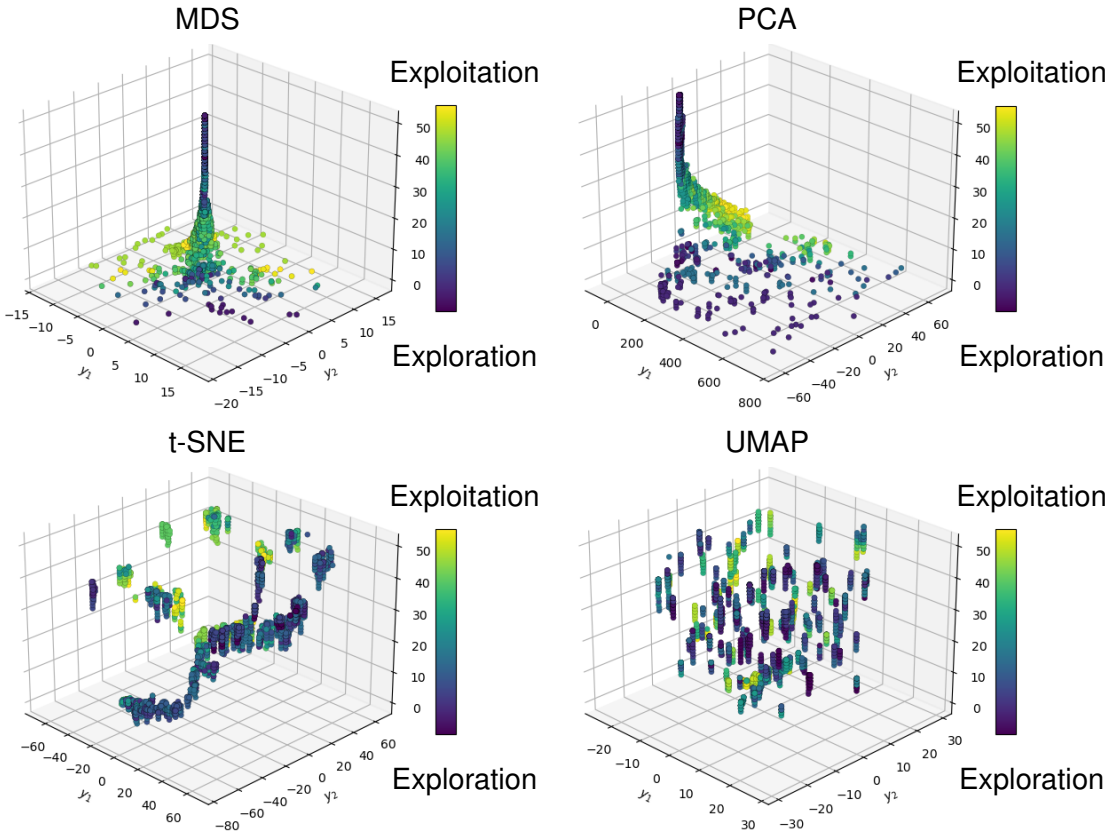


Figure 3.2: Solutions in the reduced decision space for continuous problems with $l = 5$ and $n = 1$.

correlation of the inter-generation best distances. MDS proves the most advantageous for binary problems, and MDS provides the (lowest) best RMSE value for five of the eight continuous cases. The previously cited work explicitly states that MDS is the best dimensionality technique for those tested. They note that MDS and PCA produce similar visualisation results. The t-SNE and UMAP algorithm, due to their non-linearity, rescale/magnify the denser regions of solutions - this can be seen in the recreated results of Figure 3.2. This rescaling effect of denser regions 'negatively impacts' the visualisation of population movements for this method. The paper concludes by noting that MDS takes the longest time to produce results (nearly 51 times longer to produce an MDS visualisation than a UMAP visualisation). The complexity of MDS, $O(N^3)$, limits the practical use of this technique for problems requiring a larger number of solutions, e.g., a larger population size and/or for problems requiring more generations. For benchmark problems like the DTLZ or WFG suite, 2500 function evaluations in many runs would not suffice to converge on the global optima. There is also no investigation of the expansion of this method to the objective space, which usually is of greater interest to the DM, given their role in choosing final solutions.

3.2 Multi- and Many-Objective Search Visualisation

The work of De Lorenzo et al. (2019) appears to show a visualisation which could be a template for developing many effective visualisations for solving a range of tasks. We first need to adapt the visualisation to overcome some limitations and increase the informativeness of the visualisation to generalise to more problems. We want to extend this visualisation for visualising the objective space along with the parameter space. The extension to the objective space is to attempt to detect problem landscape features which can be more prominent in particular spaces (e.g., for a particular problem, we may see discontinuities in the problem space, which may only be identifiable in the objective space). We also wish to visualise the objective solutions, which are as important and sometimes more desirable to visualise to the DM than the parameter space. Finally, by providing the objective space solution structure, we increase the amount of information available to the DM by allowing a comparison of the genospace and phenospace.

We also need to deal with the MDS complexity problem, which currently would not allow one to visualise benchmark problems such as the DTLZ suite in a reasonable time. Many problems require far more function evaluations than the simple problems demonstrated in the original works. With more complex and benchmark problems, we require a large number of generations or/and a large population size. So we require a methodology change to overcome the complexity issue for use with more realistic benchmark problems.

Further to the need to overcome the complexity issue, the original work only contained single objective functions. We aim to extend this to multi- or many-objective problems, which would increase the difficulty of the problem and often require a greater number of function evaluations. We adapt the visualisation technique to handle multi- and many-objective benchmark problems which contain a larger set of problem features than those used in the original work - for which we identify these specific problem features and analyse them in the results.

To create the visualisation, we first require the search history of an optimiser once the optimisation process has terminated. The optimiser can be any EA optimising over any problem. The optimiser results in a sequence of populations in which P_i is the population from the i-th generation containing information of the objective values and the decision values of each solution in the population, ranked according to its members' fitness values, where $i \in \{1,\ldots,n_{gen}\}$, where n_{gen} is the total number of generations. For the solutions in both the objective and parameter space, each set P_i, for all $i \in \{1,\ldots,n_{gen}\}$, MDS is applied to reduce the dimensionality from M or D to 2. In all cases herein, $M > 2$ and $D > 2$. The population is input into the MDS algorithm one generation at a time; therefore, the number of sequences input into the MDS algorithm is equal to n_{gen}, and the number of dimension-reduced sequences outputted by MDS is equal to n_{gen}. The resulting embedding is then used for visualisation, with the two embedded coordinates forming the x and y coordinates and the generation number, which can be obtained by the index i, providing the value for the z-axis - ultimately producing a 3-D plot.

An MDS visualisation of the decision and objective space is created for each problem.

To colour the solutions in an informative way, we attempt to illustrate the trade-off between exploration and inversely proportional exploitation. We aim to show the state the EA was operating in at that generation of the search. This was also done in the work of Črepinšek et al. (2013) and De Lorenzo et al. (2019). The exploration and exploitation values are normalised to map the values to a more interpretable range of $[0, 1]$. These values are relative to each problem and are not directly comparable; for example, comparing the raw values of this metric between very different problems may not be comparable, so more exploration could occur in one problem than in problem B. Still, it can yield a lower exploration score than problem B if the median distance for problem A is much greater than for problem B. The only exception to this exploration-exploitation trade-off colouring is the final generation which is all coloured white, and the solutions forged into a cross symbol to identify the Pareto set and front.

For the results in this work, we use a set of continuous multi-objective (three-objective) and many-objective (five-objective) test problems from the DTLZ test suite. The five test problems used are DTLZ1–4 and DTLZ7 (Deb, Thiele, Laumanns & Zitzler 2002). These five problems have real-valued decision variables lying in the region $[0, 1]$. The suggested number of decision variables, proposed in the original work of Deb, Thiele, Laumanns & Zitzler (2002), is $D = k + M - 1$, where $k = 5$ for DTLZ1, $k = 20$ for DTLZ7, and $k = 10$ for DTLZ2–4. These are scalable problems, with M being the number of objectives, $M - 1$ being the number of positional parameters and k being the number of distance parameters controlling the distance from the solutions to the non-dominating front. NSGA-II is the optimiser used for three-objective problems, whilst NSGA-III is the optimiser used for five-objective problems. The crossover probability is 0.8, and the mutation probability is set to 0.1. The distribution index, controlling the size of the perturbation, in both cases is fixed (15 for SBX, 7 for polynomial mutation). The algorithm's runtime is 100,000 function evaluations for $M = 3$ and 200,000 for $M = 5$.

3.2.1 Problem Landscape Features

One purpose of using a test suite is the diverse number of problem features in each problem. We aim to detect some of these features in the visualisation. This could then be used to detect these problem features in the MDS visualisation for novel problems with unknown landscapes. This could complement landscape analysis or be used when landscape analysis is not applicable. As we are visualising the algorithm's solutions, the problem landscape is being shown indirectly. Therefore it is not always the case that all problem features can be detected from the visualisation, at least not by a human visually.

By using test problems with known problem features, we can control the features and attempt to explicitly identify a map between the solutions being affected by the problem features and the MDS space. With this mapping, we can approximate problem features for novel problems with the caution that this mapping may not be one-to-one or always detectable. Some of the key problem features we aim to identify include: local optima, global optima, problem modality, bias and disconnected regions. We next define these terms.

Suppose a solution is optimal within all neighbouring solutions but not all solutions in the search space. In that case, it could be considered a local optimum - solutions can

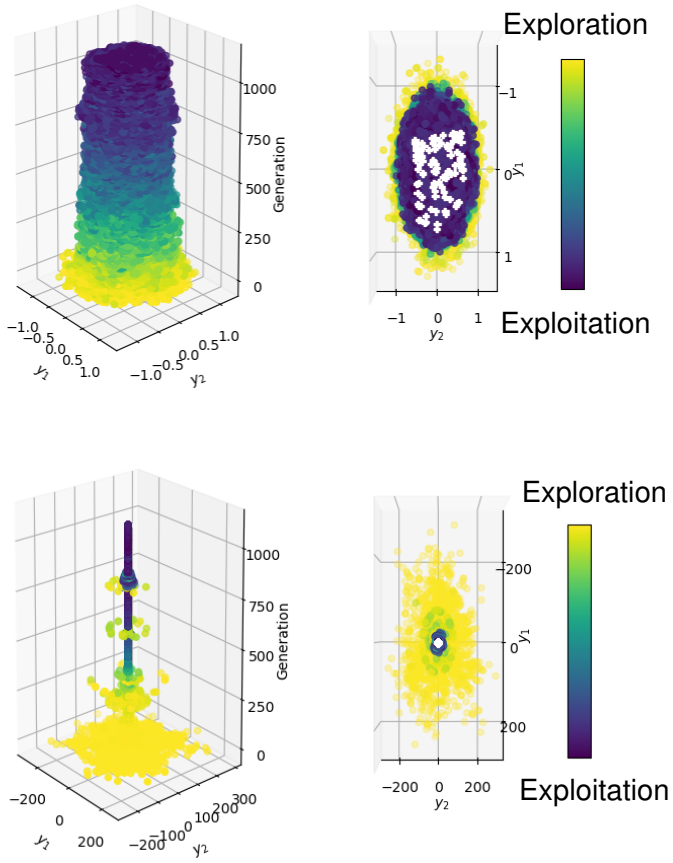


Figure 3.3: DTLZ1, with the top illustrations showing the MDS reduced decision space. The bottom illustrations show the MDS reduced objective space. The solutions are coloured according to their exploration-exploitation metric.

become trapped in regions of local optima (the solutions perturbation is unsuccessful in overcoming selection) until they overcome this difficulty by exploring outside the optima's neighbours. A solution that is optimal within all solutions possible in the search space is the global optimum. The modality of the problem may be apparent for some EA runs. The objective functions can be unimodal or multimodal. An objective function is unimodal if it has a single optimum. If the problem has multiple local optima, it is considered a multimodal problem. Problems can also contain bias. A problem is biased if there is a significant density variation of solutions in the objective space, given an even spread of solutions in parameter space. The Pareto set/front can be disconnected into regions and be considered a disconnected set/front.

3.3 Multi- and Many-Objective Problem Search Visualisation Analysis

3.3.1 Multi-Objective Results

In Figure 3.3 we visualise DTLZ1 in three objectives using the proposed methodology in Section 3.2. We can observe the MDS reduced decision space from two different az-

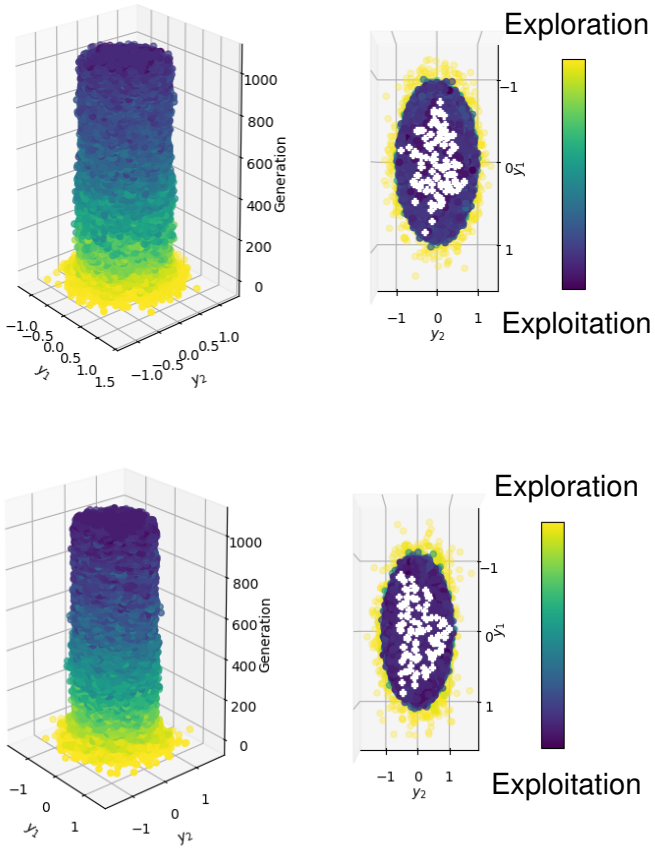


Figure 3.4: DTLZ2, with the top illustrations showing the MDS reduced decision space. The bottom illustrations show the MDS reduced objective space. The solutions are coloured according to their exploration-exploitation metric.

imuths and angles in the top row, and the MDS reduced objective space in the bottom row, again from two different perspectives. We know from the test benchmark literature that DTLZ1 is a multimodal problem and thus one of the more difficult test problems we shall consider from the DTLZ test suite for this work. As time progresses from the start of the search, we see a change in the colour indicating an initial exploration state of solution evolution converging to a more exploitative state. Furthermore, in the objective space, we see a quick change of colouring, indicating the algorithm very quickly yields an exploitative role; this is more gradual in the decision space; this can also be observed by considering the initial width of the base of the solution structure with the width of the base of the final solutions, and we can see this mapping between the decision and objective space. We can observe in the decision space the final approximation set coloured white. We can see a triangular shape and see the distributions of the approximation set. In the case of the objective space, we see the approximation front as almost a single white point because of the scale. However, when enlarging this region, we see it yields a similar structure to the approximation set and hence looks similar to the approximation front in Figure 2.2. This preservation of solution structure

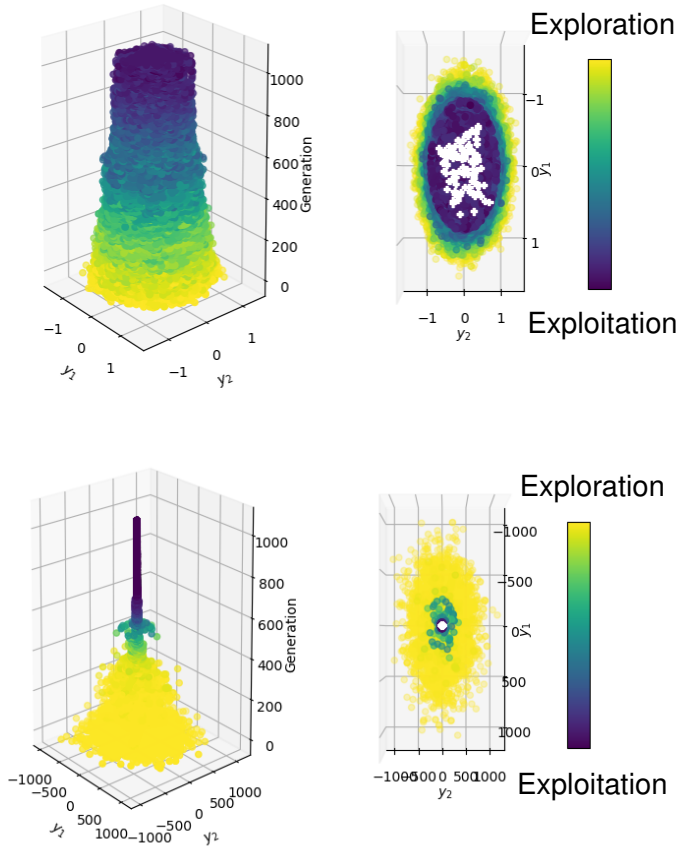


Figure 3.5: DTLZ3, with the top illustrations showing the MDS reduced decision space. The bottom illustrations show the MDS reduced objective space. The solutions are coloured according to their exploration-exploitation metric.

can be confirmed in the work of Walker, Everson & Fieldsend (2012).

One of the most striking visual artefacts of the search is the 'ring-like' structures in the objective space (bottom-left of Figure 3.3); these are the regions of difficulty for the EA. These local optima attract the solutions and hence increase problem difficulty. The ring-like structure of solutions creates this effect because the solutions are converging on both the global optima and a small subset of the population is converging on the local optima. One may expect to have seen two columns of solutions rather than a ring-like structure, which would be correct if it were not for having to apply MDS at each generation. As we apply MDS to each generation, during the MDS process, we arbitrarily rotate the solutions to orientate differently (if the principle components change), and this is why instead of seeing an additional stack of solutions, we see this ring-like structure. Although, with this knowledge, we are still able to interpret these visualisations informatively. However, this is not completely optimal as, if a problem has multiple optima, we may wish to see multiple columns of solutions converging rather than a denser or multiple ring-like structures of solutions. We can clearly see these ring-like structures of solutions (solutions in optima) at approximately 750 generations,

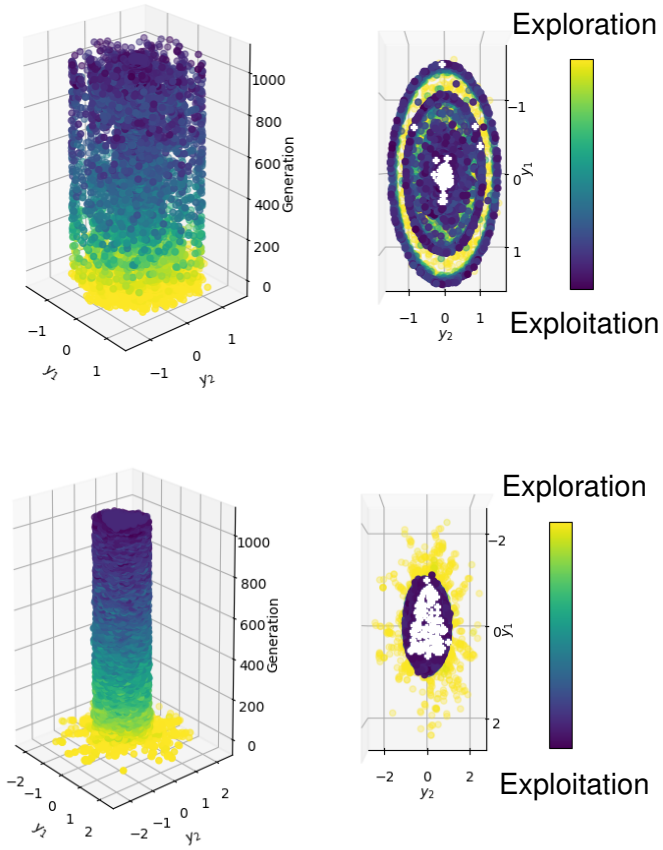


Figure 3.6: DTLZ4, with the top illustrations showing the MDS reduced decision space. The bottom illustrations show the MDS reduced objective space. The solutions are coloured according to their exploration-exploitation metric.

500 generations and a number of ring-like structures of solutions at approximately 250 generations. We see these structures in a lighter colour than the previous generations indicating that this subset of the population is moving back to a more explorative state. At generation 250, we see two prong-like structures where a subset of solutions is not only trapped in local optima but converging further away from the global optima (as MDS preserves the Euclidean distance between solutions). We can also observe an increase in the frequency of these local optima rings during an earlier stage of the search, and we can observe these solutions being trapped in optima for longer during the initial parts of the search; this is likely due to a lower initial average population fitness, and the solutions have more difficulty converging out of the optima. From these two search space visualisations, the objective space appears to be more informative, indicating a large number of algorithm/problem artefacts.

In Figure 3.4, we illustrate the solutions of an EA search on the DTLZ2 in a three-objectives problem. Unlike the previously considered problem (DTLZ1), this problem is unimodal and, as seen in the generation axis, converges much faster, demonstrating this problem is easier to converge for the same optimiser. Because of the simplicity

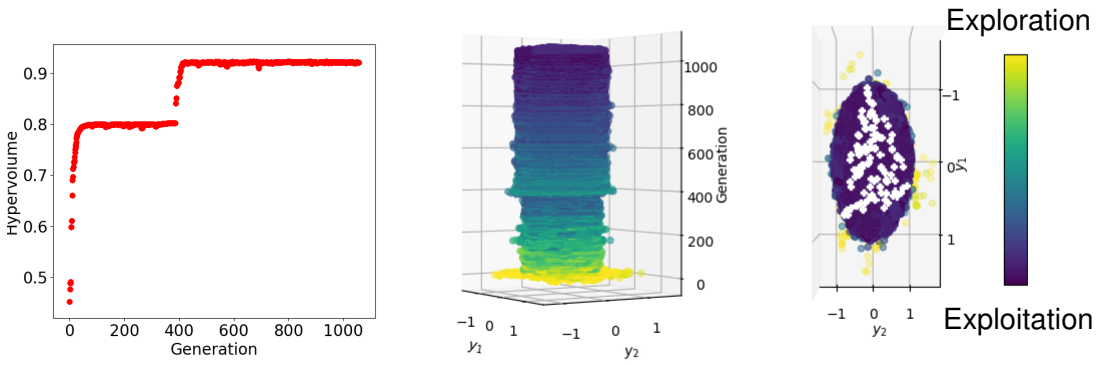


Figure 3.7: The hypervolume of a three-objective DTLZ4 problem, followed by the corresponding MDS reduced objective space plot.

of this problem, as seen in the objective space, the solutions very quickly converge to the optimum, not troubled by any local optima, and so a cylindrical column-like structure is formed. There is not a large change in scale, which is causing this cylindrical structure. We can also see a gradual colouring change indicating the gradual change from the solutions yielding an explorative state to a more exploitative one. We see in the objective space that this transition is far more gradual than DTLZ1, which is a more challenging problem. The transition through the exploration and exploitation states is also similar between the decision space and the objective space (similar progression of colouring), indicating a more similar mapping between exploration/exploitation in the decision space and the effect of exploration/exploitation in the objective space, this is unlike the DTLZ1 problem visualisation. We can also see the structure in both spaces as relatively similar, with the approximation set forming a square-like pattern (slightly compressed by the MDS process) and the approximation front structure showing a triangular shape approximation front, very similar to the Pareto front in Figure 2.2 but compressed into two dimensions. As this problem is very symmetrical, the issue of rotation caused by the MDS arbitrarily rotating the solutions is not significantly degrading any informativeness of the visualisation; this effect is caused during the selection of linear principle components aiming to maximise the preservation of Euclidean distance similarity between the dimension reduction mapping.

Figure 3.5 shows the MDS reduced solutions for an EA executing on the DTLZ3 problem in three objectives. Like DTLZ1, this problem is multimodal and, as can be seen, more challenging than the previous unimodal problem at converging to the global optima. We can see a less gradual increase in the exploration to exploitation state from the metric colouring; this is particularly prominent in the objective space where there is a fast, near-instant transition to a more exploitative state. We see this difference between the decision and objective spaces, meaning small exploration-like changes in the decision space do not always correlate to a mapping of small exploration (exploitation) in the objective space. We can also see local optima at around generation 500; this local optima is the inverse shape of the outward and upward facing prongs as seen in DTLZ1, resulting in the solutions in local optima converging to the global optima rather than away from the global optima.

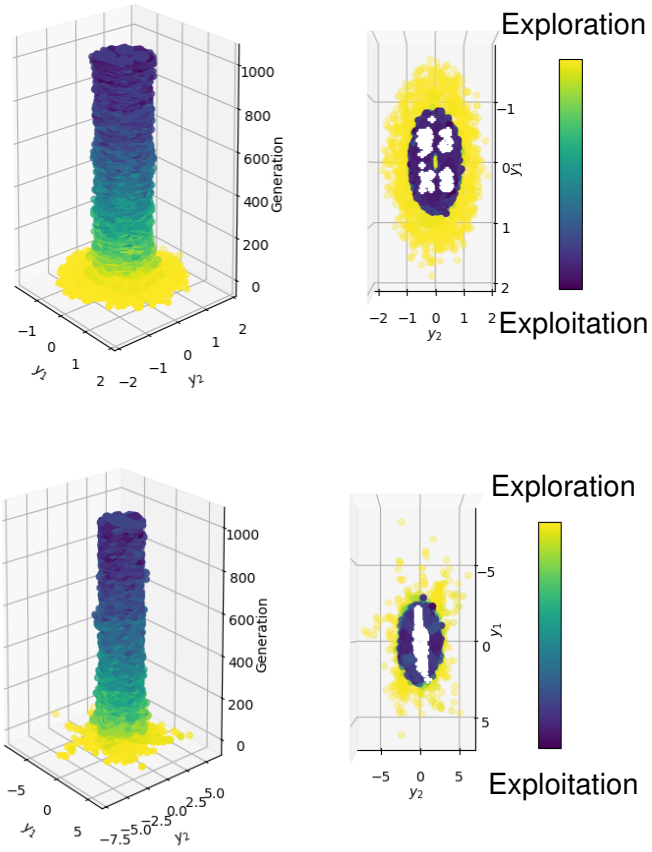
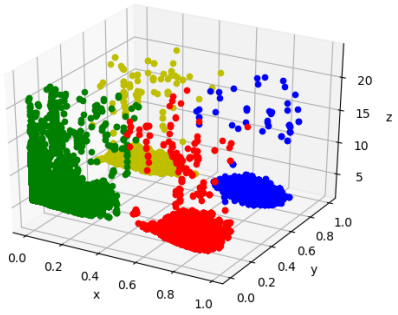


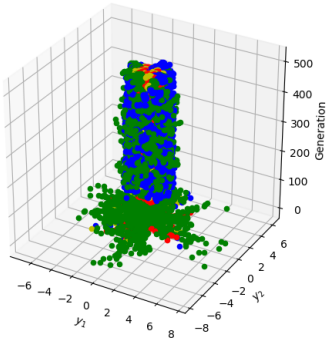
Figure 3.8: DTLZ7, with the top illustrations showing the MDS reduced decision space. The bottom illustrations show the MDS reduced objective space. The solutions are coloured according to their exploration-exploitation metric.

Compared to DTLZ1, the solution does not become trapped in local optima as frequently, but the initial base of solutions remains much larger for longer, indicating a more challenging start of the search process for DTLZ3 at converging on the global optima. For DTLZ3, the solutions do not appear to transition into a more exploitative stage of the search until around generation 500, unlike DTLZ1 where this occurs around generation 250. The solutions in DTLZ1 for this run become trapped in local optima, seen as ring-like structures of solutions, in the latter stages of the search more often, and so DTLZ1 appears to show a more difficult search in the later stages of the search whilst DTLZ3 appears more difficult in the search in the earlier stages for this particular search instance. Like DTLZ1, the positive orthant of a unit sphere shaped approximation front is preserved and can be seen when zoomed into the white point of solutions and cross-checked with Figure 2.2. Although this has been compressed to two dimensions from three, the structure and shape of the approximation set/front are clear from the visualisation.

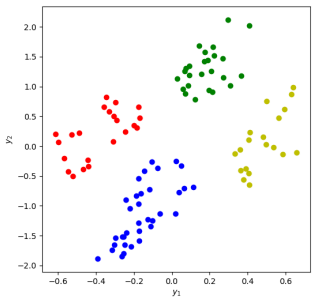
In Figure 3.6, we can see the solutions to the DTLZ4 problem optimised in three-objectives. We can see in the parameter space a cylindrical structure similar to DTLZ2.



(a) Decision space



(b) Decision space search history



(c) Final generation

Figure 3.9: Clustering coloured MDS reduced decision space. For Subfigure 3.9a, axes x, y, z correspond to the three objectives. In Subfigure 3.9b and Subfigure 3.9c, axes y_i correspond to the reduced MDS data axes.

However, we see additional rings of solutions around the structure. The DTLZ4 problem contains a bias in the (f_1, f_M) plane, increasing the difficulty of the problem; this is because the search space contains a dense area of solutions next to the (f_1, f_M) plane. It is likely that this bias is creating these ring-like structures as a subset of the solution focus on these areas of bias during the search. This bias is not visible from the objective space visualisation because the bias has not yet been overcome. This example also shows the ring-like solution structures at the termination stage of the search indicating to the decision maker that further search is required to overcome this bias and converge to the global optima.

Figure 3.7 provides a further example for which the bias has been overcome. This bias is evident from the objective space visualisation where a sudden divergence and then convergence exists, as seen from the sudden increasing width of the cylindrical solution structure. We have also provided a hypervolume graph to confirm these results, both indicating a strong convergence at generation 400. One of the key benefits of this visualisation, rather than using the hypervolume alone, is that one can see the individual solutions and determine which subpopulation and which areas of the search space the EA is finding challenging. We can see the size of the subpopulation in difficulty. We can again see the approximation set and approximation front from the visualisations.

Figure 3.8 shows the dimension-reduced solutions for DTLZ7 in three objectives. One of the most striking observations seen in the decision space is the disconnected ap-

proximation set in white. We should see a similar disconnected approximation front in the objective space, but the MDS process has compressed some of this structure, so it is not entirely obvious. In this case, the parameter space appears more informative than the objective space. The base of the decision space structure appears to indicate exploration for a greater number of generations than as seen in the objective space, noting the mapping between the two spaces. The DTLZ7 problem is a mixed modality problem. Objectives $f_1, \ldots, f_{M-1}$ are unimodal and objective f_M is multimodal. Overall the EA appears to converge to the global optima quickly, and the Pareto set can be seen as white solutions in the final gen with the majority of the population converged into the correct structure.

In Figure 3.9, we colour the solutions based on one of the four clusters. We see how the decision space population appears to converge to the global optima in four pillars (Subfigure 3.9a). This is not as clearly seen in the parameter space MDS reduced visualisation because of the arbitrary rotation issue caused by the MDS process (Subfigure 3.9b). The final generation can be seen in (Subfigure 3.9c).

3.3.2 Many-Objective Results

We now consider the many-objective problems which tend to be more difficult for the EA to solve as well as evaluating the information preservation effect of the dimension reduction process of MDS. As a greater number of dimensions is reduced, more information is lost in the process, and so the clarity of the results becomes more fogged. Using many-objective problems also means we need to use a larger number of function evaluations so the solutions can converge on the global optima for these more difficult problems. In these many-objective problems, we use 200,000 function evaluations which also results in a greater amount of time to compute and illustrate.

In Figure 3.10 we see the five-objective DTLZ1 problem in the parameter and objective space. We can observe the fast transition from an explorative state to an exploitative state in the objective space; this transition occurs abruptly at approximately generation 100, and this is supported by the exploration-exploitation trade-off metrics colouring. We can also see a ring-like structure of solutions in the final generations, indicating a subset of the solutions are in local optima; for an unknown problem landscape, one may wish to continue running this EA in case the EA has encountered a deceptive front and the potential for better solutions could exist. In the decision space, we can observe a slightly less clear Pareto set structure, and this is because the dimension process causes a greater loss of solution structure information than in three objectives as less of the distance variance is captured.

In Figure 3.11, we see the solutions converging on DTLZ2 in five objectives. For the DTLZ2 problem, we see the approximation front shape as the positive orthant of a unit sphere; therefore, when the approximation front has been compressed to two dimensions from five dimensions, the shape of the approximation front is a five-edged polygon. To generalise this result, an M-dimensional DTLZ2 problem will compress to a M-edged polygon until the dimensions increase so much that the information lost is too great and the edges of the shape become indistinct - then the shape takes a cylindrical form. In terms of the exploration-exploitation trade-off metric, we see a smooth transition through the states. Even in a great number of dimensions, this problem is

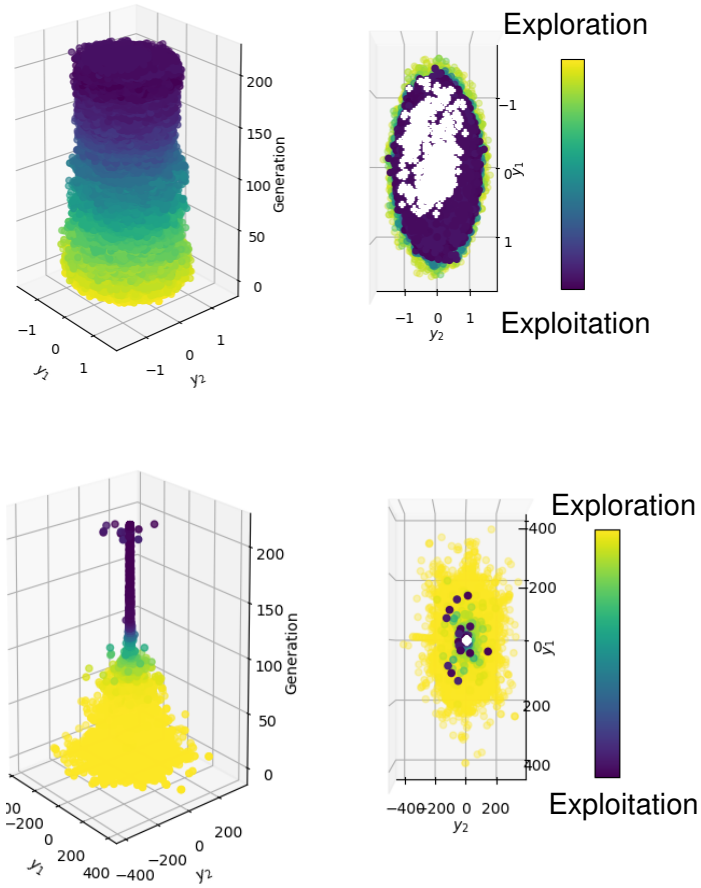


Figure 3.10: DTLZ1, with the top illustrations showing the MDS reduced decision space. The bottom illustrations show the MDS reduced objective space. The solutions are coloured according to their exploration-exploitation metric.

relatively easy for the EA to converge on the global optima, and we can see this from the smooth transition of exploration/exploitation states and the smooth cylindrical near-featureless structure of solutions. Again, we see this cylindrical type structure partially brought on by the arbitrary rotation stage of MDS.

For the solutions converged on DTLZ3 in five objectives, as seen in Figure 3.12, we see an increasing base width of solutions initially in the search then later in the search, the base width of solutions decreases as the Pareto set evolves over the generations and changes its shape. We see from the exploration-exploitation trade-off colouring that there is not an exact mapping between the states of the decision and objective space. We see the Pareto set and front not producing the same clarity of structure as seen in the three-objective problems, likely due to the increased difficulty of the problem and the larger reduction in dimensions and hence loss of greater information.

In Figure 3.13, we see the solutions converging on DTLZ4 in five objectives. In the

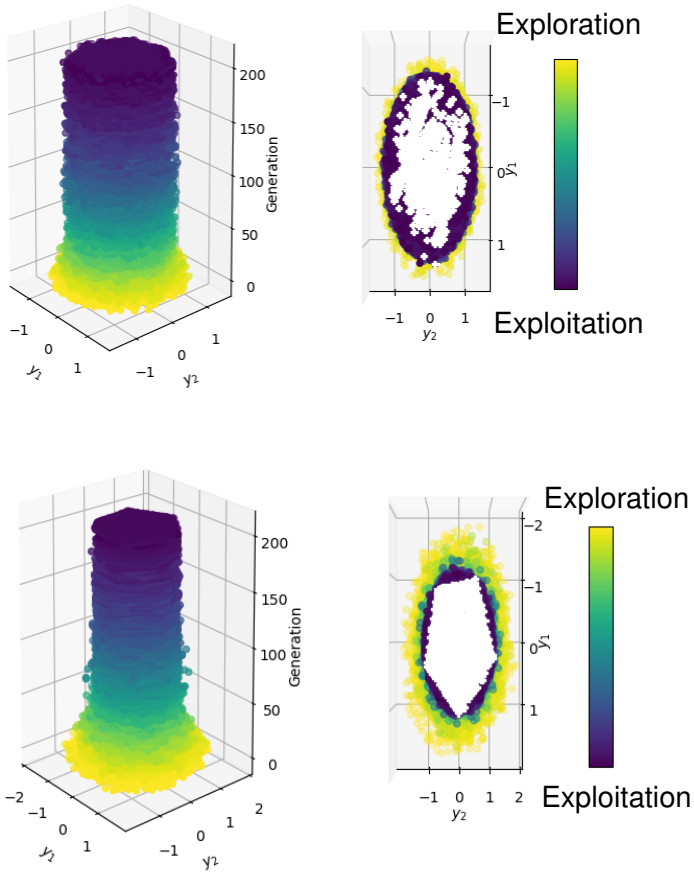


Figure 3.11: DTLZ2, with the top illustrations showing the MDS reduced decision space. The bottom illustrations show the MDS reduced objective space. The solutions are coloured according to their exploration-exploitation metric.

decision space, just like the three-objective DTLZ4, we see the solution ring structures indicating a subset of the population converging in a different region than the solutions converging to the global optima. We know this problem contains a bias (Deb, Thiele, Laumanns & Zitzler 2002) which increases solution convergence difficulty to the global optima. It is likely there is a small subset (fewer solutions making up the ring-like structure of solutions) in this five-objective problem as the solutions will be spread around a greater area of the search space due to the exponentially increasing search space as the dimensionality of the problem increases. Like DTLZ2 in five objectives, we can also see the Pareto front form a pentagon-like structure as expected from the five objectives. We see from the exploration-exploitation trade-off metric colouring that the transition from exploration to exploitation appears to be smooth.

In Figure 3.14 we see the five-objective DTLZ7 in the parameter and objective space. Compared to DTLZ7 in three objectives, we no longer see the four-pillar Pareto set structure, but we see a number of pillars which makes the Pareto set structure difficult

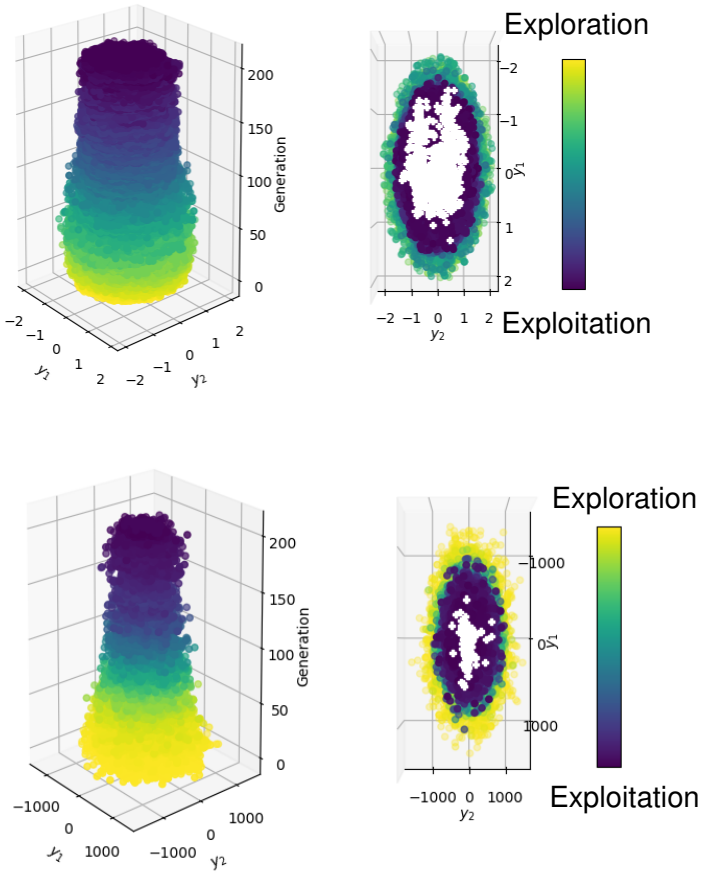


Figure 3.12: DTLZ3, with the top illustrations showing the MDS reduced decision space. The bottom illustrations show the MDS reduced objective space. The solutions are coloured according to their exploration-exploitation metric.

to observe. The same effect can be seen in the objective space where we see 'lines' forming on the cylindrical structure; these are the pillars of solutions compressed by the MDS process reducing the dimensions by taking linear components of the solutions in their original objective space.

3.4 Visualising Population Dynamics to Examine Algorithm Performance

In Section 3.3, we demonstrated the effectiveness of the method adaptation for producing visualisations of the population in the decision and objective space. This adaptation has allowed one to apply this visualisation to other more common test problems with a much greater number of objectives and larger populations which before would not have been possible due to the complexity of MDS and the larger number of function evaluations required for optimising challenging problems. We have also seen how the visualisation can indicate interesting features of the problem and algorithm artefacts via the algorithm's performance. Visualisations of different problems can be compared

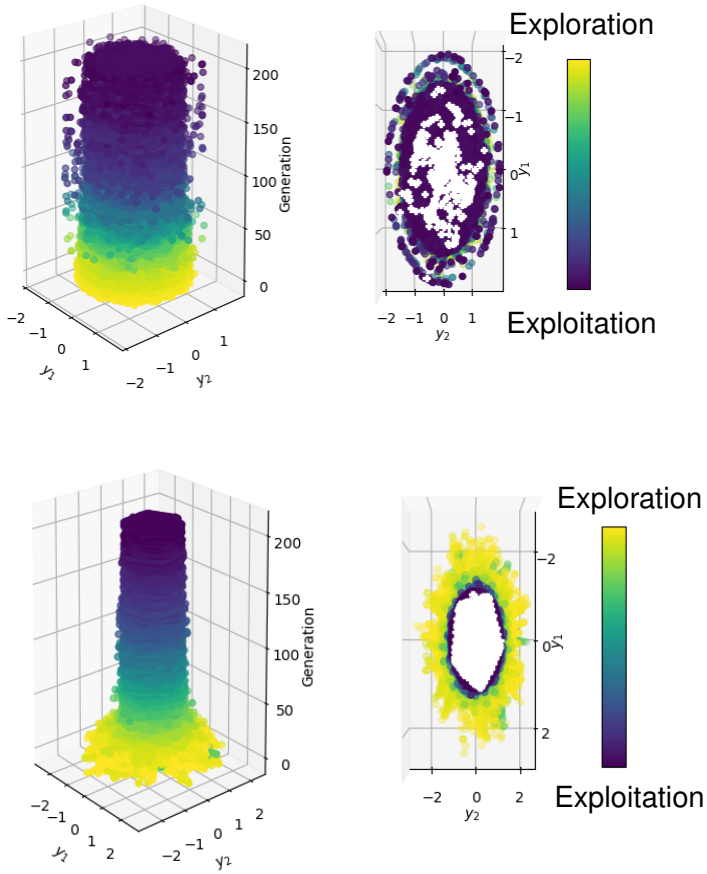


Figure 3.13: DTLZ4, with the top illustrations showing the MDS reduced decision space. The bottom illustrations show the MDS reduced objective space. The solutions are coloured according to their exploration-exploitation metric.

to enhance insight into the search. A general mapping obtained for future unseen problems allows one to identify how the algorithm performs and reveals information about the problem space.

Whilst we have identified from the visualisation areas where the algorithm encounters local optima and challenges, it is not always clear to see the positions of solutions relative to the solutions in previous or future generations over time. We have seen this from the ring-like structure caused by a subset of the population and the arbitrary rotation of MDS; during the MDS stage, linear components are chosen to preserve maximum variance/distance between solutions. So whilst the strengths of this method (time and the preservation of solution structure/indication of algorithm performance) and weaknesses (difficulty comparing the solution positions through time) are clear, we next work to develop a method to get a balance of all three strengths. This must be a low-complexity method that can preserve solution structure and project in a way for which solution positions can be compared over time. This is important to bring a

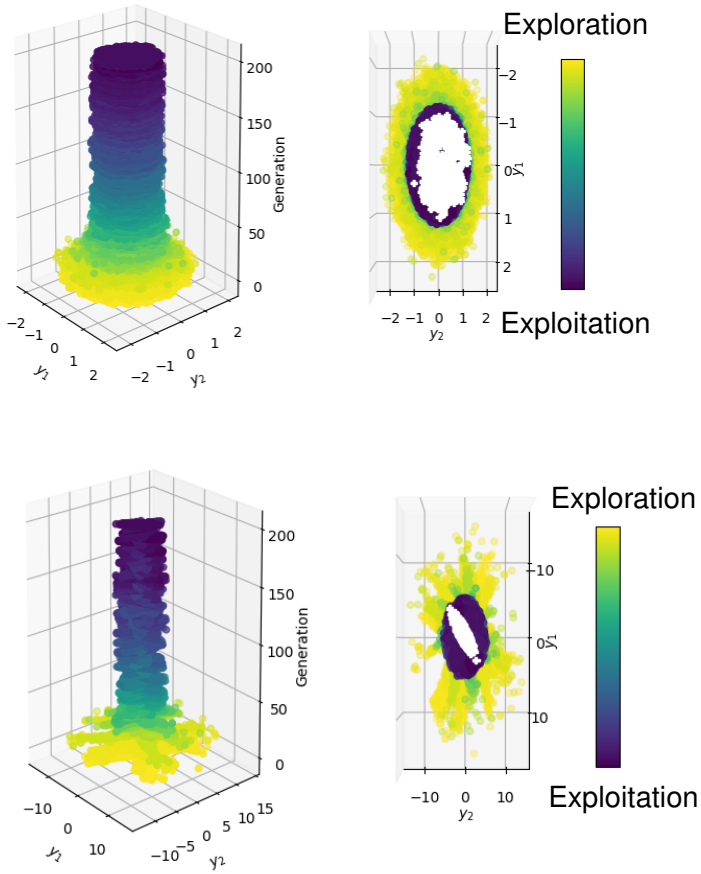


Figure 3.14: DTLZ7, with the top illustrations showing the MDS reduced decision space. The bottom illustrations show the MDS reduced objective space. The solutions are coloured according to their exploration-exploitation metric.

more practical and informative visualisation of the evolutionary search process. With better visualisations, EA researchers, developers and users are more likely to consider these visualisations to understand and represent their EAs, illustrating a clearer, more informative picture of the evolutionary search process.

3.5 Landmark Multi-Dimensional Scaling for Visualising Search

A way to create an EA population front visualisation that balances algorithm complexity with accuracy is to use an MDS approximation method for the dimensionality reduction process. We, therefore, consider replacing MDS with Landmark Multi-Dimensional Scaling (LMDS) (De Silva & Tenenbaum 2004) and evaluate this visualisation. We describe the methodology and provide detail on LMDS in the following subsections.

Just like the previous visualisation, we visualise the population, and we apply a dimension reduction technique to configure a more visually manageable 2-D representation of the population in the embedded space, which can be plotted in a series of scatter

plots. There are many different data reduction techniques which may be implemented for the purposes of visualisation. In this work, we chose Landmark Multi-Dimensional Scaling, which is an approximation method for MDS, and we aim to approximate MDS based on the results of De Lorenzo et al. (2019) which examines the efficacy of preserving the population structure for single-objective problems. MDS is also known to preserve solution structure in multi- and many-objective populations (Walker, Everson & Fieldsend 2012). In this visualisation, we do not perform LMDS at every generation but instead perform LMDS on the whole population at the end of the EA run. The benefits of doing this prevent solutions from misaligning at different generations due to the arbitrary rotation of choosing linear components orientated differently due to the different solution structures. Unlike MDS, which has too high complexity, LMDS makes this process achievable even for large function evaluations.

3.5.1 Landmark Multi-Dimensional Scaling

It is important to motivate the adaptation to LMDS before detailing the methodology. MDS has limitations. A known limitation is that MDS does not preserve 100% of the information during the reduction process. However, the results of this work and others (Walker, Everson & Fieldsend 2012, De Lorenzo et al. 2019) have demonstrated MDS to be sufficient at preserving a suitable amount of information required to reflect the non-dominated set and the search process accurately. Another major limitation when considering the application of MDS to a large EA population is its high complexity. A significant computation time and a substantial amount of RAM are required to store a Euclidean distance array (often many gigabytes for the DTLZ (Deb, Thiele, Laumanns & Zitzler 2002) test suite requirements, particularly for many-objective problems). When considering the memory complexity, classical MDS requires $O(N^2)$ storage, where N is the population size. Classical MDS also has a time complexity of $O(CN^2+N^3)$, where C is the cost of computing and accessing each entry of $\mathbf{D}$.

The MDS time complexity bottleneck is the PCA process which for k vectors can be computed in $O(kN^2)$ time using the power method (Golub & Van Loan 1996), and so as N increases, the computation time becomes impractical for use with EA population visualisations. To overcome the complexity issue, maintain population structures, and arbitrary rotation issue, we use a modified form of MDS termed Landmark MDS (LMDS) De Silva & Tenenbaum (2004). This method is an approximation method, so a trade-off is made between accuracy and time. As LMDS has not been used to visualise populations or solutions in an EA domain, it is also important to compare it to existing techniques like MDS. We show in the results that we provide a good trade-off by maintaining a very similar accuracy with a large reduction in time. The results of this work will therefore show this also to be a good visual approximation for generating informative visualisations for EA populations. Considering the complexity of LMDS, for landmark MDS, the required Euclidean distance matrix storage is $O(nN)$ and the computation requires $O(CnN+knN+n^3)$ time, where n is the number of landmarks and k is the dimensionality of the input matrix.

Compared with the complexity of MDS, we see a significant improvement in complexity, cubic to linear in N, which shall practically lead to lower, and now appropriate computation times, for the visualisation. For more on LMDS complexity, see the work of De Silva & Tenenbaum (2004) which proposes LMDS in a machine learning domain. The bot-

tleneck of MDS is in the eigenvalue decomposition, PCA stage, of the $N \times N$ proximity matrix. LMDS overcomes this complexity challenge by only utilising a subset of the entire population matrix, rather than the whole population like MDS, for eigenvalue decomposition. This subset is referred to as the set of landmark points, and the remaining solutions use a faster computing distance-based triangulation procedure to decide their position. This results in a lower complexity method due to a smaller number of calculations passing through the bottleneck process, and the rest of the calculations are made by a much lower complexity approximation/alternative.

To detail the LMDS methodology, to implement LMDS we need to designate a subset of n landmark points from similarity matrix $\mathbf{D}_N$. There are different ways to choose these landmarks, but for this work, we arbitrarily choose the landmarks from the population which performs sufficiently well; however, there may be benefits to choosing landmarks based on population distributions or certain regions of the search space, but this is beyond the scope of this work. We create matrix $\mathbf{D}_n$ and apply MDS to this $n \times n$ matrix as we described with classical MDS in Section 3.1.3 to create a $k \times n$ matrix $\mathbf{Y}_K$. After embedding the landmark points in $\mathbb{R}^k$, we now need to embed the remaining points using the landmark points (which have now been dimensionally reduced), and an affine linear transformation. We use distance-based triangulation:

1. Let $\mathbf{d}_i$ denote the ith column of the squared proximity matrix $\mathbf{D}_n^{(2)}$ for $i = 1, \ldots, n$, where each column is the vector of squared distances from the i_{th} landmark to all landmarks.
2. Compute the mean of the columns: $\mathbf{d}_\mu = (\mathbf{d}_1 + \mathbf{d}_2 + \ldots + \mathbf{d}_n)/n$.
3. With the eigenvalues and eigenvectors obtained in the previous steps calculating $\mathbf{Y}_k$, we then compute the pseudoinverse transpose $\mathbf{Y}_k^\#$ of $\mathbf{Y}_k$ with the formula

   $$\mathbf{Y}_k^\# = \mathbf{e}_i^T / \sqrt{\lambda_i}$$

   for all $i = 1, \ldots, k$.
4. To position a non-dimensionality reduced remaining point $\mathbf{X}_a$, compute the embedding vector $\mathbf{X}_a = -\frac{1}{2}\mathbf{Y}_k^\#(\mathbf{d}_a - \mathbf{d}_\mu)$, where $\mathbf{d}_a$ is the vector of squared distances between the point $\mathbf{X}_a$ and the n landmark points.

It is important to note that for a k-dimensional embedding we require at least $k+1$ landmarks. In Algorithm 1 we provide pseudocode of the LMDS process from the original LMDS work.

3.5.2 LMDS Visualisation Methodology

To develop the visualisation, we execute an EA on the problem for a set number of function evaluations, often until the algorithm has converged to the Pareto set/front or until we reach a pre-defined generation that we are interested in visualising. In the case of this work, we terminate the EA early in some cases to focus on interesting artefacts of the problem or algorithm. During the EA evaluation process, we save the objective space population coordinate data (the objective values) and the generation to which that solution belongs. Once the EA has reached a terminal state and we

Algorithm 1 The LMDS process pseudocode. The code was extracted from the original LMDS work by De Silva & Tenenbaum (2004).

procedure LMDS PROCEDURE
    Input:
    $N \leftarrow$ number of data points;
    $\Delta \leftarrow$ squared-distance function (two arguments);
    $\mathrm{n} \leftarrow$ desired number of landmark points;
    $k \leftarrow$ desired number of output dimensions;

    Step 1: select landmark points (randomly).
    $\mathrm{P}(1:\mathrm{N}) \leftarrow \mathrm{randperm}(\mathrm{N})$; ▷ random permutation
    $\ell(:) \leftarrow \mathrm{P}(1:\mathrm{n})$;

    Step 2: classical MDS on landmarks.
    for ($i = 1:\mathrm{n}, j = 1:n$) **do**
        $\Delta_n(i,j) \leftarrow \Delta(\ell(i),\ell(j))$;
    for $i = (1:n)$ **do**
        $\vec{\mu}_\mathrm{n}(\mathrm{i}) \leftarrow \mathrm{mean}\,(\Delta_\mathrm{n}(i,:))$;
    $\mu \leftarrow \mathrm{mean}\,(\vec{\mu}_\mathrm{n}(:))$;
    for $(i = 1:n, j = 1:n)$ **do**
        $\mathrm{B}_\mathrm{n}(i,j) \leftarrow -\left(\Delta_\mathrm{n}(i,j) - \vec{\mu}_\mathrm{n}(i) - \vec{\mu}_\mathrm{n}(j) + \mu\right)/2$;
    $[\mathrm{V},\lambda] \leftarrow$ eigensolve (B_n, k); ▷ $V(i,j) = i$-th component of j-th eigenvector
    ▷ $\lambda(j) = j$-th eigenvalue, in descending order
    $k_+ \leftarrow \min(k$, number of positive eigenvalues);
    for $(i = 1:k_+, j = 1:n)$ **do**
        $L(i,j) \leftarrow V(j,i) * \mathrm{sqrt}(\lambda(i))$;

    Step 3: distance-based triangulation.
    for $(i = 1:k_+, j = 1:n)$ **do**
        $L^\sharp(i,j) \leftarrow V(j,i)/\,\mathrm{sqrt}(\lambda(i))$;
    for $(j = 1:N)$ **do**
        $\mathrm{X}(:,j) \leftarrow -\mathrm{L}^\sharp * \left(\Delta(\ell(:),j) - \mu_\mathrm{n}(:)\right)/2$; ▷ matrix-vector multiplication

    Output:
    $\mathrm{k}_+ \rightarrow$ actual number of output dimensions
    $\mathrm{L} \rightarrow \mathrm{k}_+$-dimensional embedding of landmarks
    $\mathrm{x} \rightarrow \mathrm{k}_+$-dimensional embedding of data

have collected all the objective space population coordinate data (the objective values) and the corresponding generation value, we can apply LMDS to the objective space population coordinate data to reduce the population dimensionality from M to two dimensions. This frees up the third (z) dimension, which we use to project the population in the z-axis based on the generation it belongs to, giving a 3-D plot. We can then colour the plot according to some metric to add additional information.

In this work, visualisations are coloured depending on the state in which the algorithm is operating, either exploring the search space or exploiting an area of high quality in the search space. To do this, we use the exploration-exploitation trade-off metric detailed in Section 3.1.2. The colouring could also be based on other metrics such as hypervolume or other indicators such as generation number. However, for this work, we feel exploration-exploitation trade-off colouring is more informative. It is important to note the limitations of the methods one implements, and so we note there exist limitations to the exploration-exploitation trade-off metric, such as the raw values themselves not being significantly informative (hence not being included in the visualisations) and the colouring is problem dependent. Furthermore, for this visualisation, as with the previously discussed visualisation in Section 3.2, we colour the non-dominated solutions as white crosses.

We adapt LMDS for EA population visualisation by choosing landmark points that contribute significant value to preserving the MDS solution structure in the LMDS space. We, thus, ensure the landmark points are uniformly randomly chosen solutions from all generations to produce an equal distribution of solutions from across the generations for the context of visualising the EA population rather than limiting the landmark selection from the initial generation or choosing completely random landmarks as suggested by De Silva & Tenenbaum (2004) (used in a data science setting), which could produce an unequal distribution of solutions from different generations in the context of visualising the EA population.

For the visualisations in this work, we chose to visualise the objective space population, although the same method would allow one to visualise the decision space in the same way. There is existing literature to suggest visualisation of the decision space is sparse (Schäpermeier et al. 2020), and one could produce visualisations using this method for the decision space.

3.6 Analysis of Landmark Multi-dimensional Scaling for Visualising Search

In this section, we analyse the results. As with the previous visualisation method, we aim to identify a problem feature mapping from the objective space to the MDS/LMDS reduced space. With this method, the solution positions can be more easily compared between generations but possibly at the cost of less accurate results because of the implementation of the approximation method. We investigate both of these factors with qualitative (graphical interpretation) and quantitative (statistical) analysis. We start by analysing the three-objective problems to compare the MDS/LMDS reduced plot with the objective space to detect the key problem and algorithm mappings. Once the mappings are understood, we apply the knowledge to analyse many-objective problems.

3.6.1 Experimental Parameters

The problems are optimised with NSGA-III (Deb & Jain 2013) unless stated otherwise with a crossover probability set as 0.8, and the mutation probability is set to 0.1. NSGA-III has been shown to produce similar results when benchmarked against NSGA-II on multi-objective problems; however, NSGA-III performs significantly better on many-objective DTLZ test suite problems for both convergence and diversity measures (although they note NSGA-II can outperform NSGA-III for some instances such as many-objective knapsack problems (Ishibuchi et al. 2016)). Nonetheless, the implementation of the most optimal algorithm or parameter choices is not the focus of this work.

The experimental parameter for the algorithm used in the rest of this chapter is as follows: The distribution index is fixed at 15 for SBX and 7 for polynomial mutation; these values control the perturbation size. The number of divisions in the boundary layer used in the automatic generation of reference points is set to the default parameter value of 12. Note that since the algorithm termination condition is checked after each iteration, it can be possible for the algorithm to exceed the predefined function evaluations (by a maximum of a single population size). The function evaluations used for each problem are defined in the figure caption. The number of function evaluations varies for each problem depending on the difficulty of the problem. If a problem is more difficult, it will require a greater number of function evaluations to converge. Therefore, the number of function evaluations correlates with the difficulty of the test problem implemented. However, to demonstrate certain artefacts of the search or features of the problem, we sometimes terminate the search early, which will be indicated in the caption of the figure. We also extend the problem set to include the WFG problem suite to detect more problem mappings with a more diverse problem set. Next, we discuss the configuration of the evaluation problems.

DTLZ parameters

The evaluation phase comprises of operating an EA (with the previous paragraph's configurations) on four continuous problems from the DTLZ test suite (Deb, Thiele, Laumanns & Zitzler 2002); these problems are DTLZ1, 2, 4 and 7. These problems have real-valued decision variables lying in the region $[0, 1]$. The suggested number of decision variables is $p = k + m - 1$, where $m - 1$ is the number of position parameters and k is the number of distance parameters, controlling the distance the solutions are from the Pareto front. In this work, $k = 5$ for DTLZ1, $k = 20$ for DTLZ7, and $k = 10$ for the remaining problems. The problems have a mixture of different fronts. The problems are scalable in the number of parameters and objectives; in this experiment, three- and five-objective problems are utilised.

WFG Parameters

The second suite of problems used in this work is the WFG test suite problems (Huband et al. 2006); by extending the test problem set used for evaluation with problems from other suites which contain additional features, we can identify further mapping than with using the DTLZ problem suite alone. All the WFG problems used in this work have a concave-shaped Pareto front, with the exception of the WFG convex problem (Figure 3.16). The problems in the WFG suite vary in modality. To configure the multi-modal

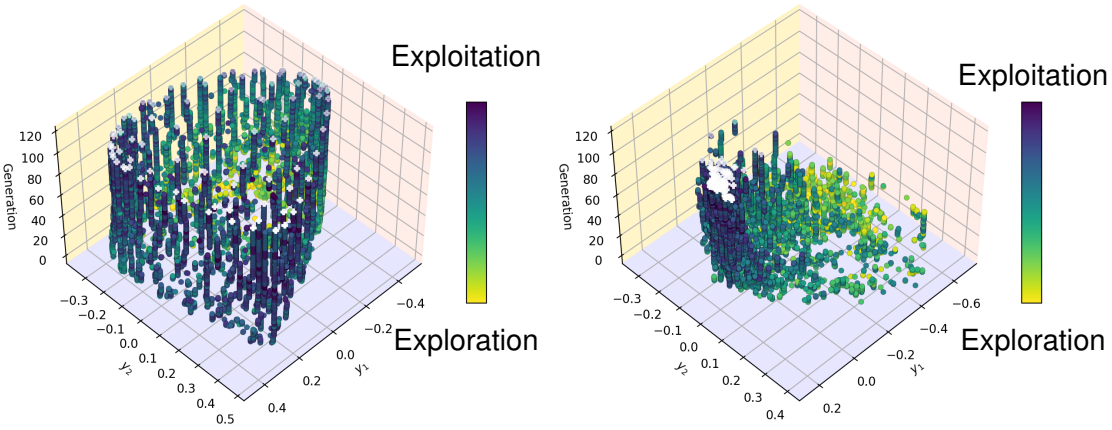


Figure 3.15: LMDS reduced objective space (left) and decision space (right) of a WFG concave problem in three objectives produced with 5000 landmarks and 10,000 function evaluations.

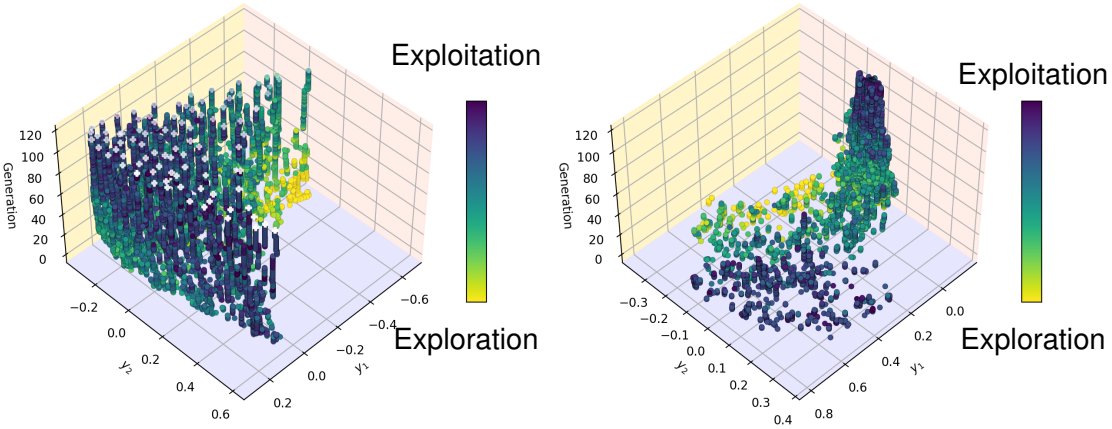


Figure 3.16: LMDS reduced objective space (left) and decision space (right) of a WFG convex problem in three objectives produced with 5000 landmarks and 10,000 function evaluations.

problems used in this work, we set the number of optima, controlled by transformation functions (Huband et al. 2006), to 6 optima (5 local optima and 1 global optimum), and the global optimum is located at 0.35 in the first objective, with a magnitude of 2.

3.6.2 Qualitative Analysis

Multi-objective Results

We now aim to identify mappings between the objective and parameter space to their respective reduced spaces. With these mappings, we aim to transfer this knowledge when evaluating other problems to identify features of the problems. To identify a mapping, one would first need to implement the visualisations on a problem suite in which the user can control the quantity and type of features. We, therefore, developed test problems using the WFG test suite (Huband et al. 2006), creating unimodal and mul-

timodal problems where all other problem features are identical in order to create a mapping identifying a feature of interest. The WFG suite was designed as a set of functions which provide different features to develop test problems. The authors also provided a series of examples, namely WFG1-9, created using their provided functions. However, in most cases of the literature where WFG is utilised, most authors do not create their own problems with these WFG functions but instead use the WFG1-9 examples provided. This level of control allows one to create tailored test problems motivating the use of WFG for this work. We create a series of multi-objective (3 objectives) test problems with minimal features except for the ones we wish to identify for mapping from objective to manifold embedding space. All problems must have a Pareto shape function. For each EA evaluation, we produce two figures: the figure positioned on the left of the page is the reduced objective space, and the figure positioned to the right of the page is the reduced parameter space visualisation. We see solutions coloured with their exploration-exploitation trade-off metric, and we see white solutions in the final generation to represent the approximation front/set.

In Figure 3.15, we created a unimodal concave front problem by utilising the concave Pareto shape function. We can now identify a mapping from the objective space to the reduced space with concave Pareto fronts. We first note that the shape and orientation from low-dimensional to lower dimensional space are preserved, and we can identify the Pareto front geometry. We notice the solutions are denser on the boundaries of the front and less dense in the centre of the Pareto front structure. This logically seems correct if one considers the 3-D Pareto front being compressed into 2-D. For the same test problem but with a convex Pareto front, seen in Figure 3.16, we can see the opposite effect; the Pareto front appears to contain a denser region of solutions in the centre of the structure and fewer solutions on the outside/boundary of the structure. We note the Pareto set and front geometry appear to be preserved generally with LMDS.

Compared to the Subsection 3.2 methodology, the arbitrary rotation problem is mitigated, producing a clear trail of solutions as they evolve over time. For example, we can now see many columns of solutions showing the journey of the individual solutions to either the Pareto front or their termination. We see many of the initial solutions, x, y positions are nearly identical to the final x, y positions, indicating a simple and unchallenging problem for the EA to optimise. Figure 3.17 shows a concave multimodal problem; we can see this to be a more challenging problem with many columns of solutions starting and stopping at different times of the search. The solutions can fall into regions of local optima and create columns of solutions which disappear as the solution is perturbated to search space with better fitness. This creates additional challenges for the EA and can keep solutions in local optima regions for longer periods than regions which do not contain optima, affecting the rate of convergence to the global optima. We notice in the more challenging WFG problem the solutions exhibit a darker colouring, indicating lots of exploitation throughout the search.

In Figure 3.18, we see DTLZ1; comparing it to the Subsection 3.2 methodology, we can again obtain a much clearer understanding of the solution path due to the mitigation of the arbitrary rotation issue. We can see the path of solutions leading to global optima. We notice the solutions form a triangle shape with a dense region of solutions forming the boundary; this is an artefact of NSGA-III. Because NSGA-II and III always select the

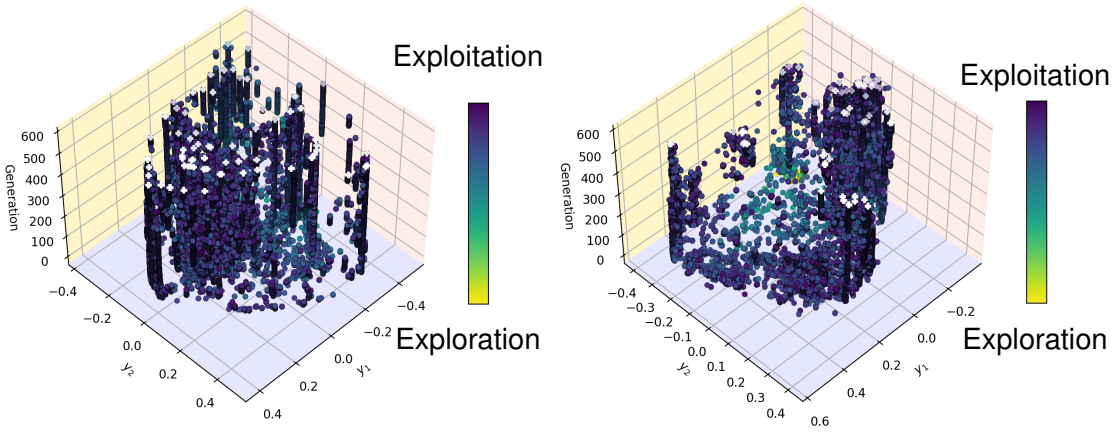


Figure 3.17: LMDS reduced objective space (left) and decision space (right) of a WFG multi-modal problem in three objectives produced with 5000 landmarks and 50,000 function evaluations.

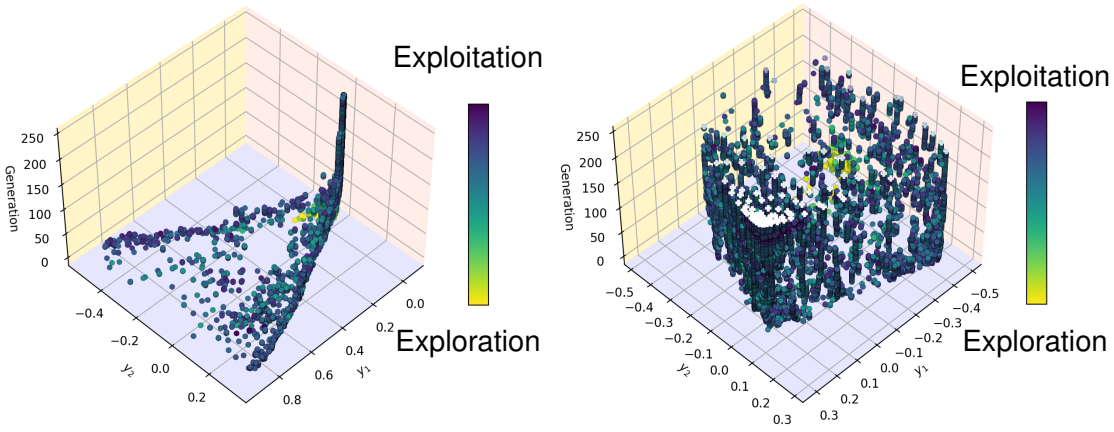


Figure 3.18: LMDS reduced objective space (left) and decision space (right) of DTLZ1 in three objectives produced with 5000 landmarks and 20,000 function evaluations.

solutions at the further edges of the search to maintain diversity during the selection operation, we see a dense area of solutions at these edges as these solutions are always chosen for the next population. In contrast, solutions in the centre region of the search are not always chosen as these solutions do not have the greatest diversity (in terms of the spread of solutions), creating this distribution effect. By visualising the distribution of solutions, we could also notice areas of interest not explored by the EA, and so we could create a weighted reference plane to direct solutions to other more promising areas of the search. We also see a small white point of solutions at the top of the column of solutions converging to the global optima. If we were to zoom into this region, a more triangular shape approximation front could be seen like the one in Figure 3.19, DTLZ2.

We can see that compared to figure DTLZ2, the initial solutions are not as close to the

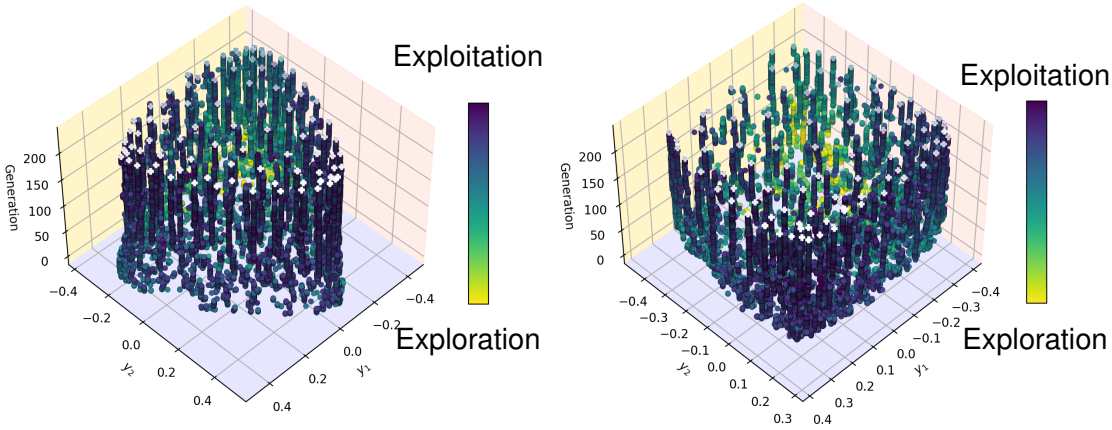


Figure 3.19: LMDS reduced objective space (left) and decision space (right) of DTLZ2 in three objectives produced with 5000 landmarks and 20,000 function evaluations.

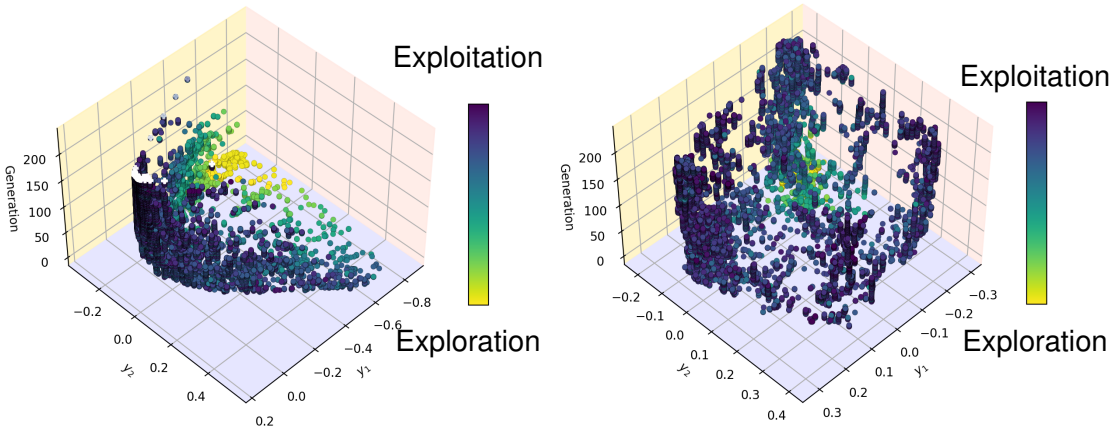


Figure 3.20: LMDS reduced objective space (left) and decision space (right) of DTLZ3 in three objectives produced with 5000 landmarks and 20,000 function evaluations.

final approximation set, and hence they create a scalability effect (where the DTLZ1 Pareto front geometry can only be seen by zooming in). This is because DTLZ1 is a harder problem than some of the more simple unimodal counterparts of the DTLZ suite, for example, DTLZ2. In the parameter space, we can map the denser regions of the space to the denser regions of the objective space to see which areas of the parameter space correlate to which areas of the objective space. From these DTLZ figures, we can see that both DTLZ1 (multimodal) and DTLZ2 (unimodal) MDS reduced visualisations preserve much of the Pareto and the population geometry. We can compare these figures to identify mappings. In Figure 3.19 (DTLZ2), the MDS reduced visualisation is very similar in structure to the WFG unimodal problem (Figure 3.15). No columns of solutions are formed away from the global optimum, and the solutions appear to converge to the global optimum almost instantly. This showed that

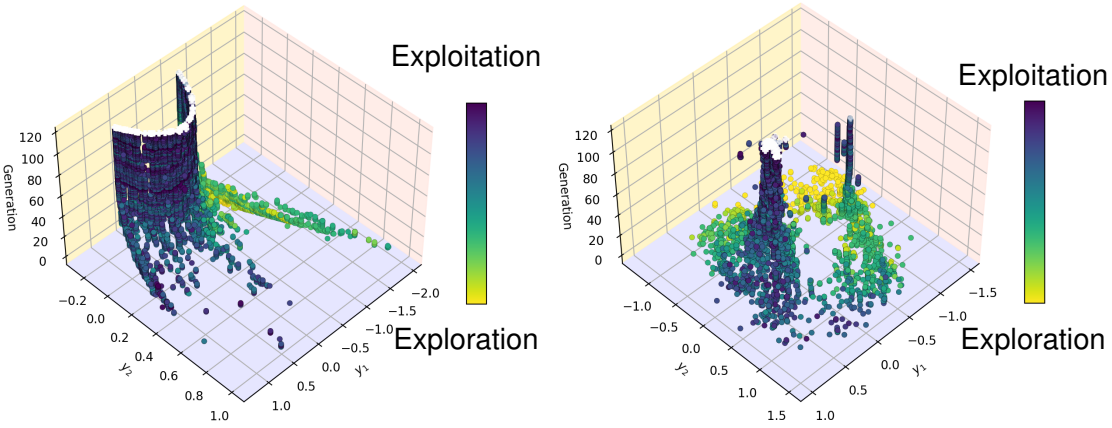


Figure 3.21: LMDS reduced objective space (left) and decision space (right) of DTLZ4 in three objectives produced with 5000 landmarks and 10,000 function evaluations. The EA process has been terminated early to demonstrate population movements.

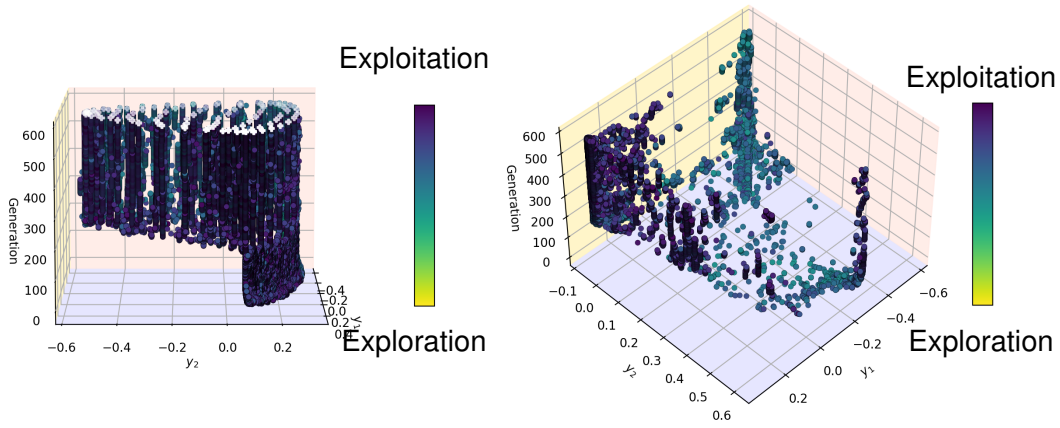


Figure 3.22: LMDS reduced objective space (left) and decision space (right) of DTLZ4 in three objectives produced with 5000 landmarks and 50,000 function evaluations.

the algorithm had encountered little difficulty converging to the global optima.

DTLZ3 in 3 objectives can be observed in Figure 3.20. Compared to the previous two evaluated problems, we see it is most similar to the DTLZ1 problem as it appears to search a wide range of the search space initially before converging on a small area of the search space. DTLZ3 is, like DTLZ1, a multimodal problem and, as seen, more challenging to converge to the optima than DTLZ2. From the many solutions in local optima outside the global stack of solutions, we can see that for this particular instance, DTLZ3 appears more challenging than DTLZ1, which converges much quicker to the global optima (at approximately generation 100). From the parameter space, we see less dense regions of solutions, such as the middle of the search space, with denser regions of solutions in the corners of the search.

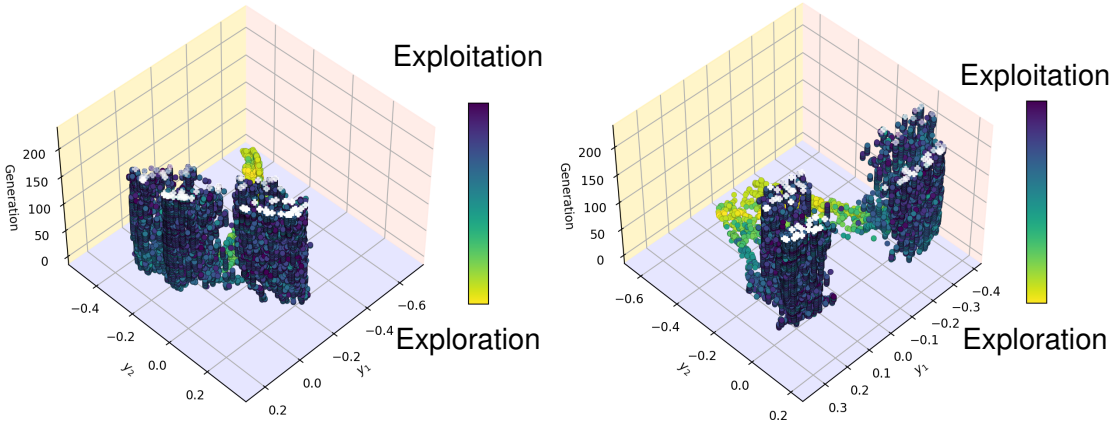


Figure 3.23: LMDS reduced objective space (left) and decision space (right) of DTLZ7 in three objectives produced with 5000 landmarks and 20,000 function evaluations.

In Figure 3.21, we see DTLZ4 terminated early to focus on an artefact of the problem, and in Figure 3.22, the full evaluation of DTLZ4 overcoming this artefact. This artefact is a bias. DTLZ4 has bias in the (f_1, f_M) plane: in this case, (f_1, f_3). This bias can be seen in the reduced objective space of Figure 3.21, and we see in Figure 3.22 the bias of the problem overcome by the solutions at approximately generations 200-300. We can see the paths of the solutions to the global optima; these are the solutions kept by the NSGA-III reference plain diversity operator. We would expect to see the overcoming layer exhibit more exploration after an early stage of exploration, but this is not indicated in this example; we would expect to see a lighter layer of solutions at the overcoming generation showing some of the limitations of the exploration-exploitation metric.

In Figure 3.23, we see DTLZ7 in three objectives; in the original objective space, we usually see four disconnected regions of solutions, and we see these four clusters in the reduced objective and decision space. As we see for unimodal problems, we often see n-structures of solutions for n-disconnected fronts, with the odd solution which has been perturbed to a new area of the search space but does not remain in the population for the next generation due to weak fitness. We can see from the exploration-exploitation metric a gradual transition through the two phases.

Many-objective Results

We now want to consider if these mappings still hold when evaluating many dimensional problems and reducing the solution dimensions from many to two dimensions with LMDS for visualisation. The DTLZ test suite is scalable to many objectives, so we extend the previously considered problems DTLZ1–4 and DTLZ7 to five objectives. Because the number of objectives increases and the NSGA-III reference plane value is set at 12 solutions across the edge of the M-dimensional plane, the cardinality of each population increases by default and is larger than the three-objective population; this is to compensate for the non-linear increase in search space.

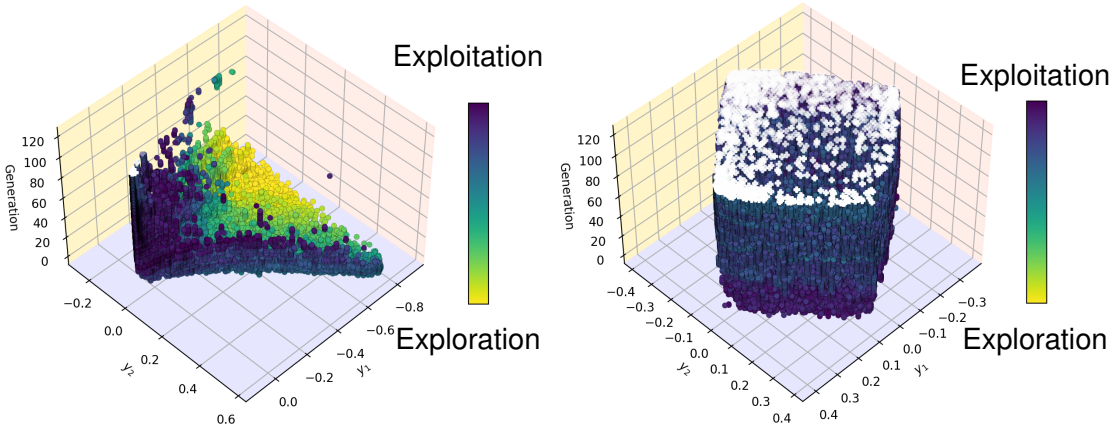


Figure 3.24: LMDS reduced objective space (left) and decision space (right) of DTLZ1 in five objectives produced with 5000 landmarks and 200,000 function evaluations.

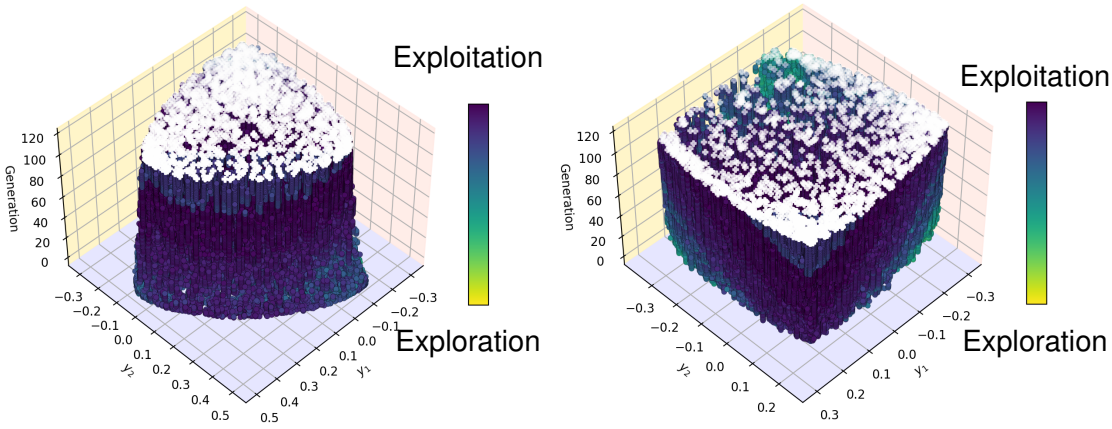


Figure 3.25: LMDS reduced objective space (left) and decision space (right) of DTLZ2 in five objectives produced with 5000 landmarks and 200,000 function evaluations.

In Figure 3.24, we observe DTLZ1 in five objectives in the reduced objective (left-most figure) and parameter space (right-most figure). We notice the LMDS visualisation of the search generates a similar geometry to the three-objective case. We can see solutions stuck in regions of local optima, and in this five-objective case, solutions appear to remain in regions of local optima more frequently; this is expected given that the increase in difficulty is correlated with increasing objectives. To reaffirm this, we can also see many more solutions still not converged to the global optima even at the final generation of the search. The solutions converge more sporadically than the smoother transition seen in the three-objective variant. In the reduced parameter space, we see the solutions constrained to a square shape, whereas in the original space, they lie in the region $[0, 1]$ in each of the five objectives.

In Figure 3.25, the reduced objective and parameter spaces of DTLZ2 in five ob-

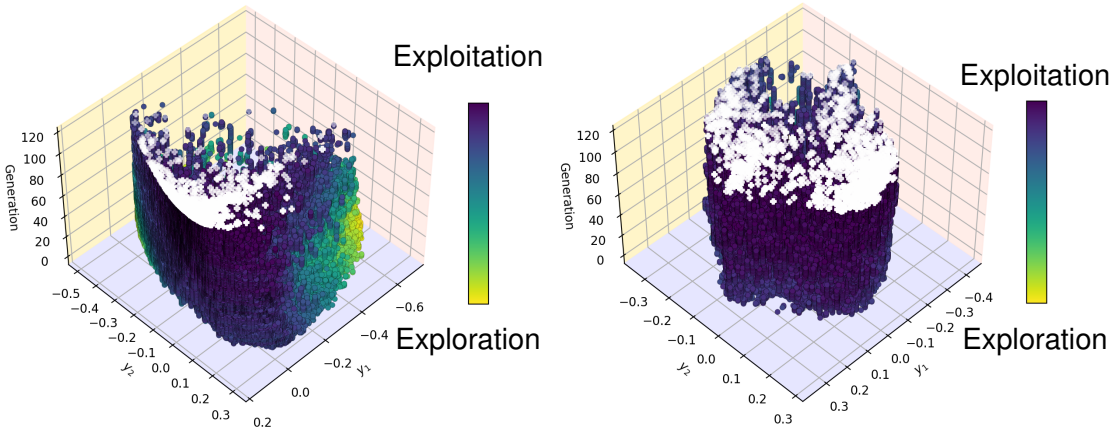


Figure 3.26: LMDS reduced objective space (left) and decision space (right) of DTLZ3 in five objectives produced with 5000 landmarks and 200,000 function evaluations.

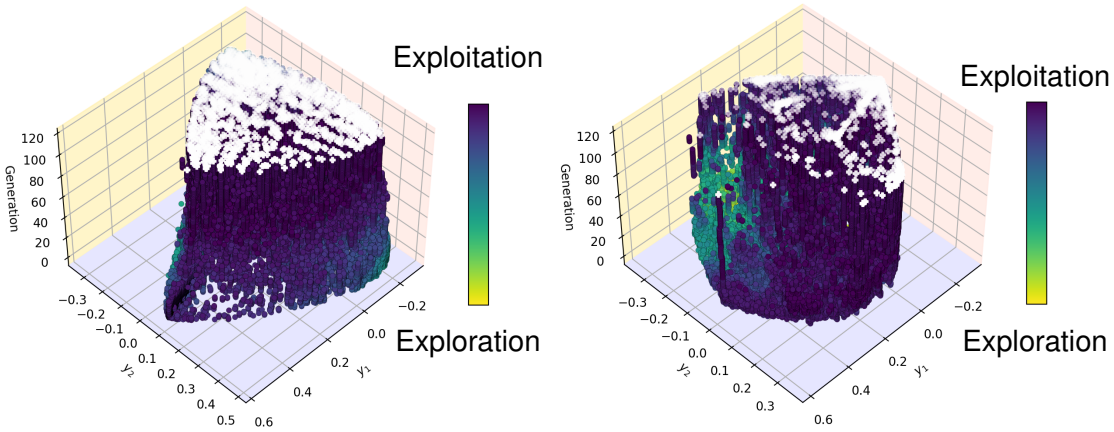


Figure 3.27: LMDS reduced objective space (left) and decision space (right) of DTLZ4 in five objectives produced with 5000 landmarks and 200,000 function evaluations.

jectives can be seen. We see the solutions exhibit a similar geometry to the three-objective problem. We see the same concave approximation set geometry as in the three-objective problem. Unusually for this problem, one would usually expect to see a pentagon shape approximation set; this is due to there being five objectives and the NSGA-II and III diversity operators always choosing the solutions at the ends of all objectives. It may be because the largest perturbations were made in only three dimensions during most of the search. Like DTLZ2 in three objectives, we see smooth continuous paths of solutions to the optima indicating this is a relatively simple (compared to multimodal DTLZ problems) unimodal problem.

The five-objective DTLZ3 problem visualisation can be seen in Figure 3.27. Like DLTZ4 in three objectives, it is a challenging multimodal problem with local optima present. As with the three-objective variants, we can see that the EA struggles to converge to the

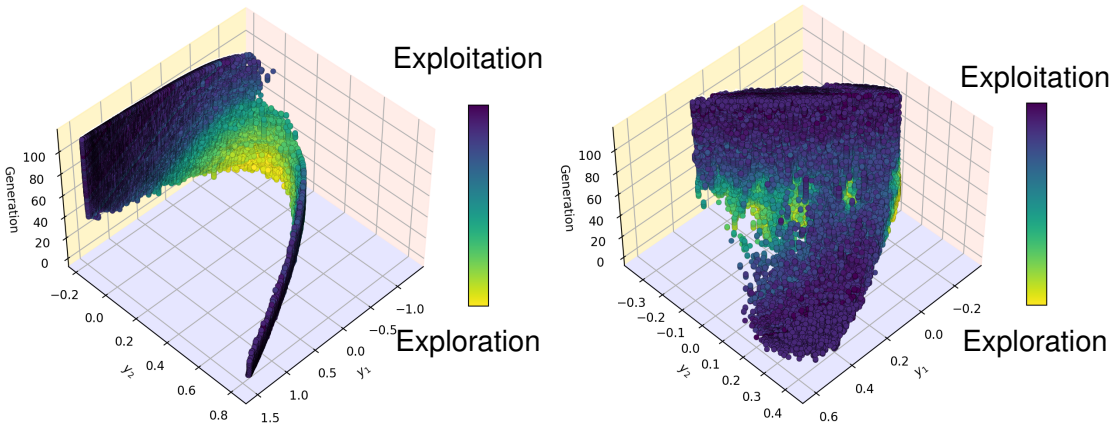


Figure 3.28: LMDS reduced objective space (left) and decision space (right) of DTLZ7 in five objectives produced with 5000 landmarks and 200,000 function evaluations.

optima much more than DTLZ2 and DTLZ1. Although most solutions appear to be in the correct region of the global optima, the approximation front geometry does not yet look converged, signifying that this instance may require a longer run time to converge. In DTLZ4 seen in Figure 3.27, we see that unlike in the three-objective variant, the bias was overcome very quickly, perhaps because of the stochastic nature of EAs, or the increase in objective space may have provided more available search space, which may increase the chance of overcoming the bias.

Finally, DTLZ7 in five objectives can be seen in Figure 3.28. This looks similar to DTLZ7 in three objectives with a similar trajectory of solutions to the optima. However, we can see the expected geometry of the approximation set is not easily seen as clusters appear more compressed, making them more difficult to detect because of a more substantial compression of information from five dimensions to two dimensions than from three dimensions to two dimensions. We also notice that the colouring metric indicates an exploitative stage, followed by an exploring stage and then reverts back to an exploitative stage. Many of the same mappings can be identified in the many objective examples but, in some cases, are more difficult to detect because of the increase in information loss during the dimension reduction. We can generalise and say that as the number of dimensions to be reduced increases, the information loss increases, and the mappings become increasingly more difficult to detect visually.

Algorithm Performance Testing

We next consider varying the algorithm on the same problem and considering this visualisation as a way to examine algorithm and parameter performance. The visualisation is of DTLZ1 in three objectives with 5000 landmarks and 30,000 function evaluations. The algorithms considered are NSGA-II (Deb, Pratap, Agarwal & Meyarivan 2002), NSGA-III (Deb & Jain 2013), MOEA/D (Zhang & Li 2007) and SPEA2 (Zitzler et al. 2001), NSGA-III with a random mutation operator and NSGA-II with crossover and mutation both set at a value of 1. We also provide the hypervolume in the figure caption for comparison. The results of this experiment can be seen in Figure 3.29.

Figure 3.29: LMDS reduced objective space of DTLZ1 in three objectives. Top left algorithm is NSGA-II, with hypervolume 0.8202. Top middle algorithm is NSGA-III, with hypervolume 0.8171. Top right algorithm is SPEA2, with hypervolume 0.7117. The bottom left algorithm is MOEA/D, with hypervolume 0.7541. The bottom middle algorithm is NSGA-III with a 'ruin and recreate' mutation operator and hypervolume 0. The bottom right algorithm is NSGA-II with mutation (PM) and crossover (SBX) parameters of 1, and hypervolume 0.6748.

The most effective algorithm under consideration, which can be seen as the algorithm whose solutions make a smooth transition to the Pareto front, quickly overcoming local optima, is NSGA-III. From the visualisation, NSGA-III appears to encounter only a few local optima up to generation 50. Consequently, we could tune more optimal parameters for this algorithm evaluation, although in comparison with the other tested algorithms, it is the most effective algorithm with its default parameters. We see NSGA-III overcome optima faster than SPEA2 and NSGA-II, as, in the non-global optima regions, columns of solutions appear for shorter periods of time. We can see by generation 50 that all solutions have converged close to the global optima. We can also see the algorithm artefacts, like we have seen in the previous examples, detecting the mappings; for example, solution density increases along observable paths showing the algorithm artefacts of the crowding distance (for NSGA-II) or reference plane (for NSGA-III) selection acting on the search. Comparing with the provided hypervolumes, we can see how many of these problems and algorithm artefacts cannot be detected by the hypervolume or other common metrics like IGD alone because of the inability to consider the search at the individual solution level, which LMDS can achieve, providing a much more informative representation/examination of the search than these metrics alone.

The next most effective algorithm tested is NSGA-II which can be clearly identified from the evaluation of algorithm LMDS visualisations. We can see in the z-axis that NSGA-II converges to the global optima in fewer generations than SPEA2, but a larger subsection of the population encounters difficulty than in the previous NSGA-III example; this can be seen as a more coarse transition of solutions to the region of global optima with solutions in non-global optima regions stalemating in these regions for longer periods that the solutions of better optimisers.

We then evaluate SPEA2, placing it in a similar category to NSGA-II regarding algorithm performance when comparing the visualisations. For SPEA2, we see much more of the subpopulation trapped in local optima than in the previous algorithms. We also see the solutions trapped in regions of optima further away from the global optima than other previous algorithms considered, and the solutions are trapped for more generations (longer columns of solutions trapped in regions of local optima). These indicators from the visualisation all show SPEA2 to be less effective than NSGA-II and III for this instance. From the colouring metric and the position change from the initial to final solutions, MOEA/D appears to demonstrate very little exploration and the solutions yielded an exploitative state throughout the whole search. Interestingly, this is an artefact of the algorithm showing how the algorithm is working by creating sub-problems as a reference for the population to enhance convergence.

Finally, we consider NSGA-II with bad parameters configured for this problem. We set the mutation and crossover parameters values to $(1, 1)$. The resulting visualisation indicates these parameters to be poor when considered with NSGA-II with the default parameter. We see this poor performance with the solutions taking additional generations than NSGA-II (with the default parameters) to converge, resulting in more solutions converging to regions of local optima and away from the global optima than the previously evaluated algorithms.

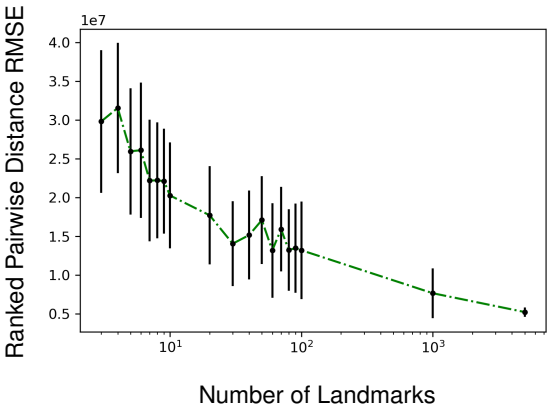
Table 3.2: Computation time, mean μ and standard deviation σ of RMSE ranked pairwise distances across 30 runs on DTLZ1 and DTLZ2 in three objectives, 10,000 function evaluations. MDS computation times were 326.37 s (DTLZ1) and 286.07 s (DTLZ2).

Problem	**Landmarks**	**RMSE** $(\mu \pm \sigma)$	**LMDS Time (s)**
DTLZ1	3	$3.07 \times 10^7 \pm 9.29 \times 10^6$	0.0112
	4	$3.24 \times 10^7 \pm 8.51 \times 10^6$	0.0018
	5	$2.68 \times 10^7 \pm 8.28 \times 10^6$	0.0021
	6	$2.70 \times 10^7 \pm 8.81 \times 10^6$	0.0022
	7	$2.30 \times 10^7 \pm 7.99 \times 10^6$	0.0026
	8	$2.30 \times 10^7 \pm 7.62 \times 10^6$	0.0027
	9	$2.30 \times 10^7 \pm 6.93 \times 10^6$	0.0031
	10	$2.12 \times 10^7 \pm 6.98 \times 10^6$	0.0037
	50	$1.49 \times 10^7 \pm 5.85 \times 10^6$	0.0202
	100	$1.46 \times 10^7 \pm 5.85 \times 10^6$	0.0366
	1000	$8.20 \times 10^6 \pm 3.75 \times 10^6$	0.6459
	5000	$5.77 \times 10^6 \pm 5.79 \times 10^5$	31.9253
DTLZ2	3	$2.26 \times 10^7 \pm 5.31 \times 10^6$	0.0051
	4	$2.08 \times 10^7 \pm 4.76 \times 10^6$	0.0018
	5	$1.84 \times 10^7 \pm 4.65 \times 10^6$	0.0022
	6	$1.68 \times 10^7 \pm 3.97 \times 10^6$	0.0020
	7	$1.69 \times 10^7 \pm 4.89 \times 10^6$	0.0023
	8	$1.52 \times 10^7 \pm 3.17 \times 10^6$	0.0025
	9	$1.40 \times 10^7 \pm 3.89 \times 10^6$	0.0024
	10	$1.28 \times 10^7 \pm 2.22 \times 10^6$	0.0028
	50	$9.31 \times 10^6 \pm 8.86 \times 10^5$	0.0186
	100	$9.33 \times 10^6 \pm 6.63 \times 10^5$	0.0357
	1000	$8.81 \times 10^6 \pm 1.57 \times 10^5$	0.6330
	5000	$8.86 \times 10^6 \pm 6.73 \times 10^4$	32.3680

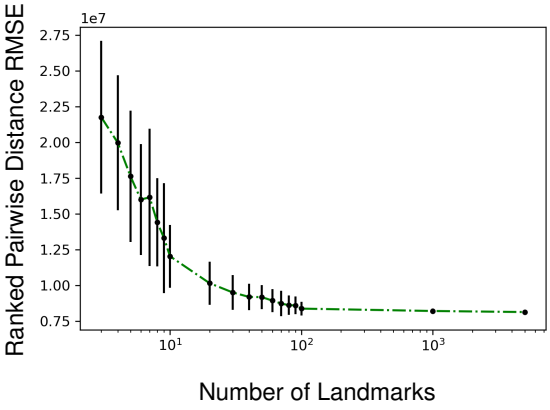
3.6.3 Quantitative Analysis

As we have seen throughout the chapter, MDS can potentially be used to provide informative visualisations of an EA search, but as the number of function evaluations increases, the complexity of MDS makes it unpractical for use with large data and hence high function evaluation EA runs; this can exclude the use of this visualisation method for many-objective or challenging problems. We have shown, however, how LMDS can be used as an effective approximator to MDS and evaluated for a series of high-function evaluation test problems showing it to be visually akin to MDS. We now consider measuring this similarity and the time taken to produce the visualisation explicitly.

LMDS applied MDS to a subset of the population, known as landmarks, and computes the remaining solution positions with a fast approximator. We established an experiment to measure the accuracy/time trade-off of LMDS to evaluate the benefit of using LMDS. In the experiment, we compute MDS and LMDS for several different configurations of the DTLZ1 and DTLZ2 test problems. We vary the number of landmarks



(a) DTLZ1



(b) DTLZ2

Figure 3.30: The MDS and LMDS ranked RMSE results for DTLZ1 (top) and DTLZ2 (bottom) in three objectives for 10,000 function evaluations.

for LMDS and compute the similarity whilst measuring the computation time. To compute the similarity, we use the Root Mean Square Error (RMSE), which measures the difference between the solutions in the MDS and LMDS space. More specifically, we use the ranked RMSE, which considers the order of solutions rather than Euclidean distance. Strikingly, one such effect of using the ranked RMSE is obtaining large numerical values owing to the large population size (approximately 10,000 solutions to be ordered). We use this to measure the order of solution positions between LMDS and MDS. We run each instance 30 times and compute the mean and standard deviation. We compute for 10,000 EA solutions. The RMSE equation can be defined as:

$$RMSE = \left(\frac{1}{n_{gen}} \sum_{h=1}^{n_{gen}} \left(\sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} \left(d_{i,j}^{M} - d_{i,j}^{L} \right)^2 \right) \right)^{\frac{1}{2}} \tag{3.14}$$

Here, n_{gen} is the number of generations, N is the population size, $d_{i,j}^{M}$ and $d_{i,j}^{L}$ are the

MDS distance matrix and LMDS distance matrix, respectively.

The DTLZ1 problem had an MDS compute time of 326.37 seconds, and the DTLZ2 problem took 286.07 seconds to compute the MDS values for visualisation. From the results of Table 3.2 and the graphs of Figure 3.30, we can see that for both problems, an increase in landmarks used decreases the RMSE. We notice this to be a non-linear relationship between the number of landmarks and the RMSE and LMDS compute time. This is a non-linear relationship for both problems; however, it appears more prominent in the DTLZ1 problem. We can also observe an obvious knee point from examining the graphs. We can therefore determine an optimal number of landmarks, which is likely problem-dependent. For the DTLZ1 and DTLZ2 problems considered in this work, the optimal number of landmarks appears to be between 10 and 100 landmarks, and so remarkably, using just 1% of the total population, we can approximate MDS, drastically reducing the time and memory required to produce these informative visualisations; for instance, for DTLZ1, the computation time of LMDS is less than 0.02% of the time required to compute the visualisation using MDS.

Finally, we compare the time complexity of all the visualisation methods (Subsection 3.1.1 methodology, Subsection 3.2 methodology and Subsection 3.5.2 methodology) in this chapter, the MDS method from Subsection 3.1.1 methodology has the complexity of $O(CN^2 + N^3)$. We ignore the coefficient terms, which are trivial given the higher exponents. Therefore, considering 25000 function evaluation (the typical number of function evaluations to solve DTLZ1 in three objectives with NSGA-II), it would compute in $(25000^2 + 25000^3) = 1.56 \times 10^{13}$ seconds. The Subsection 3.2 methodology would compute in $(250 \times (100^2 + 100^3)) = 2.525 \times 10^8$ seconds, a reduction of 99.99% of the total time. The Subsection 3.2 methodology running in parallel with 250 threads would compute in $(100^2 + 100^3) = 1 \times 10^6$ seconds. Then with the LMDS Subsection 3.5.2 methodology, the compute time with 100 landmarks is $(100^3) = 1 \times 10^6$ seconds. Overall this analysis shows a significant benefit of reduced complexity time that the two methods (Subsection 3.2 methodology and Subsection 3.5.2 methodology) proposed within this thesis bring over the method of Subsection 3.1.1 with negligible loss of accuracy for the range of problems we evaluated.

To summarise, fewer than 1% of the time and memory necessary to produce an MDS visualisation was required to produce a visually indistinguishable LMDS visualisation of the evolutionary search. This was because one only has to apply MDS to less than 1% of the population (landmarks) when using LMDS and a fast affine linear transformation approximates the non-landmark solutions. Although, when using greater than 1% of the population as landmarks, the improvement in the LMDS visualisation was negligible and difficult to observe. DTLZ2 portrays a similar story, although even fewer landmarks are required to produce a similar error as observed in DTLZ1, perhaps this owning to the symmetry of the population structure.

3.7 Conclusion

Although visualisations of the Pareto set/front are ubiquitous in the EA literature, they provide little information about the whole search process, instead focusing on a small subset of the search, the final solutions, limiting the informativeness of the visualisation. A visualisation of the whole search process could be highly beneficial and

desirable, providing a way for algorithm researchers and users to communicate, examine and understand their algorithm visually. Furthermore, EA developers may be able to demonstrate their algorithms as well as compare and evaluate the performance throughout the entire search in depth. The requirement for methods that can provide all these benefits is unquestionable and can impact and enhance the progression of the EA field.

We set out to create such a solution, starting with the examination of a promising existing visualisation of an evolutionary population. Although effective for relatively simple problems, it was no 'silver bullet' and would require adaptation and evaluation to overcome serious scalability issues in order for this to apply to most EA test problems. In the first section of this work, we recognise the scalability issues and adapt the methodology to create visualisations for complex and multi- and many-objective problems. We demonstrated the effectiveness of this method on multi- and many-objective problems, showing a number of algorithm and problem artefacts can be seen from the visualisation. Although this method was fast, allowing for scalability to more realistic benchmark problems (multi- and many-objective), and it allowed one to identify features of the problem, the local level embedding created a rotational issue for which the positions of solutions between generations was not always clear. A global level embedding would be optimal to develop a more informative visualisation for which solution positions between generations could be contrasted.

The latter section of the chapter amended this methodology to use LMDS, which is a fast approximator for MDS from the data science domain, performing a global embedding in significantly reduced computation times. We evaluated this method qualitatively (visually) to show it was much more effective at comparing solution positions between generations. Therefore more problem and algorithm artefacts could be identified and more precisely identified. We, therefore, set out to map these problem and algorithm artefacts for use with novel problems. We demonstrated the effectiveness on multi- and many-objective problems from two different problem suites. Because of the potential informativeness, we considered the visualisation as a tool for comparing algorithms and parameters to demonstrate its effectiveness. We provided a tool for algorithm developers to show the effectiveness and comparison of their novel algorithm, which as a single figure, could be used to communicate their algorithm performance in a single figure. We compared against a common metric, hypervolume, to show how LMDS provides a much greater depth of information - showing every solution in the search space, thus having a much greater capability than the hypervolume. Then, an explicit analysis was performed. We experimented with the number of landmarks, the generation time, and the accuracy of the visualisation from MDS to obtain significant results - for the benchmark problems we tested, LMDS is akin to MDS visually, whilst requiring less than 1% of the time and memory necessary to produce an MDS visualisation of the same objective space solutions.

Ultimately, this chapter provides a package of novel methods/options for EA users to potentially enhance their work via the benefits of search visualisation. It does so by aiming to address the research questions regarding EA users to communicate, examine and understand their whole algorithm search process visually. Furthermore, due to the results of this work showing the arbitrary runtime and many benefits of a potentially

high informativeness search visualisation, as well as this stand-alone visualisation, this work opens up a plethora of further applications for this visualisation (more details can be found in Chapter 4).

Chapter 4

An Explainable Visualisation of the EA Search Process

The challenges of transparency inherent to black-box EAs frequently make it difficult to understand the Evolutionary Algorithm (EA) search process, which has a bearing on algorithm development and hyper-parameter optimisation. In this chapter, we consider the importance of eXplainable AI (XAI) in evolutionary computing and based on the initial indications of high informativeness of the methodology of the previous chapter, we propose a visualisation called the Population Dynamics Plot (PopDP). The PopDP visualisation aims to enhance algorithm comprehension and the explainability of the EA search process by illustrating the population of solutions, the parent-offspring lineage, the solution perturbation operators, and the path of the search process. We apply PopDP to a series of discrete dual- and many-objective knapsack problems and real-world offshore wind farm optimisation problems to show the insight and efficacy of PopDP for visualising an algorithm search. We also evaluate the effectiveness of the PopDP against emerging benchmarking strategies for explainable AI techniques and evaluate the explainability of PopDP. This chapter material has been presented or published as three outputs:

- *Walter, M. J., Walker, D. J. & Craven, M. J. (2022), An explainable visualisation of the evolutionary search process, in 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', GECCO '22, Association for Computing Machinery, New York, NY, USA, pp. 1794–1802.*
- *Walter, M. J., Manikowski, P., Walker, D. J. & Craven, M. J. (2022), Explainable optimisation of offshore wind farms, talk at the Partnership for Research in Marine Renewable Energy 2022 (PRIMaRE).*
- *Walker, D. J., Craven, M. J., Walter, M. J. & Manikowski, P., Explaining optimisation: Applying explainable AI principles to metaheuristic, in 'Handbook of Formal Optimisation Methods', under review, Springer.*

EVOLUTIONARY algorithms (EAs) have been demonstrated with extensive literature as an effective strategy for solving optimisation problems. However, the utilisation of high-dimensional data and the complex relationships between operators and selection pressure conceal the intuition of EAs and yield EAs with *black-box* algorithm states. The hypothesis for this chapter is that rather than merely having a low-resolution overview of the search, i.e., the understanding of solutions being initialised, some obscured black-box behaviour with reproduction operators and selection to produce the output of final solutions, one may be able to gain an individual (local) solution level of understanding of the search process with explainability methods providing multiple benefits to the user.

In the year 2022, the Genetic and Evolutionary Computation Conference (GECCO)

held the first workshop to mitigate explainability issues by focusing on how XAI can support EC research and how EC can be used within XAI methods. The workshop received significant interest, with many people recognising the importance and impact of greater explainability on EC. This interest sparked a special issue of the ACM Transactions on Evolutionary Learning and Optimisation journal for explainability and evolutionary computation. The conference discussed the potential impact and benefits explainability could bring to the EA domain; this included, for example, the need for robustness, an understanding of the EA search and the problem-algorithm interaction, which would benefit algorithm users with debugging (e.g., is the model working as intended?). In addition, the conference considered the importance of determining how reproduction operators are affecting the search to support the tuning of algorithm parameters and comparison of EAs – this could be particularly useful for Evolutionary Computing developers and researchers. The conference also considered how explainability might also be able to provide traceability, which assists with an element of futureproofing. For example, if EC is used for more significant decisions, it is likely to come with greater social and legal impacts. So it could provide some accountability via traceability and potentially assist with regulation and audit compliance. Ultimately, an output of the conference was that by enhancing explainability in EC, we are also creating a greater understanding of the process, which brings more trust and hence use – people do not like to adopt technology they do not understand (Zhu et al. 2018, Pu & Chen 2006) – so explainable EC could expedite EC integration into industry just as XAI did with AI.

Further to the demands for ECXAI from researchers at GECCO, we have seen a demand for explainability within the EC literature. For example,

> "The generation of explanations regarding decisions made by population-based meta-heuristics is often difficult due to the nature of the mechanisms employed by these approaches. With the increase in use of these methods for optimisation in industries that require enduser confirmation, the need for explanations has also grown."

from the work of Fyvie et al. (2021). Highlighting the importance of explainability specifically for EA users in industry. The work of Bacardit et al. (2022) states,

> "We find that there is a growing concern in EC regarding the explanation of population-based methods, i.e., their search process and outcomes."

the work also states,

> "Recent growth in the adoption of black-box algorithms, including EC-based methods, in domains such as ... has led to greater attention being given to the generation of explanations and their accessibility to end-users"

Further, the work of Singh et al. (2022) states, "The process by which a GA arrives at a solution remains largely unexplained to the end-user. A poorly understood solution will dent the confidence a user has in the arrived at solution."

A survey for engineers was conducted in the work of Vincalek et al. (2021) to attempt to understand why the use of genetic algorithms in industry was so low despite many engineers knowing the use of EAs (65%). The results showed “76% expressed a lack of trust" in the EA solutions and generation processes. The engineers described the “blind trust" process as a “bad approach". The work concludes with

> “A key theme throughout the data is that establishing trust between engineers and their processes is important. One of the ways that this could be established is through an increase in transparency and explainability. An effective way of increasing both transparency and explainability is through visualisations.".

This work is supported by other literature highlighting the lack of trust in metaheuristics as a result of their black-box nature (Hornby & Yu 2007, Tiwari et al. 2015, Brownlee et al. 2022).

Considering the demands of the EC community and the potential advantages explainability could bring to EC, it is clear that creating better ECXAI methods will provide benefits to EA users, developers, researchers, legal, social, education and all stakeholders involved in processes utilising EAs. Better explainable evolutionary computing is important for everyone within the EC community, and in this chapter, we aim to develop one of the first explainability methods for EC. Before developing ECXAI methods, we define some key principles to enhance explainability in EC: we aim to create explainability to generate an understanding of the search space, an understanding of traceability within the search, and an understanding of the solution generation mechanisms.

In this work, we aim to address two research questions formulated on the demands/key principles listed prior. In a compact and interpretable manner (in this case, a visualisation), can we:

- Preserve the structure of the population to see the journey of the search process? Which areas of the search space have been explored/not explored? Can we observe artefacts of the problem or EA?
- Describe the inner workings of the EA? How are the solutions being generated? Which are the parents of strong solutions? Can we understand the traceability of solutions through their ancestry tree? How is the interaction between reproductive operators affecting the search? For example, is the overuse of an operator causing difficulties with convergence?

This chapter is structured as follows. First, Section 4.1 introduces the novel methodology for creating an explainable visualisation of the evolutionary search and the experimental parameters used to generate the results. Section 4.2 considers visualising the dual-objective knapsack population with PopDP. Then, Section 4.3 considers hyper-parameter analysis with PopDP. Section 4.4 then analyses the results for the many-objective knapsack problems. An analysis of PopDP on real-world offshore wind farm optimisation problems is performed in Section 4.5. We then provide a discussion on ECXAI in Section 4.6 and conclude the work in Section 4.7.

4.1 Population Dynamics Plot

As we motivated in the previous section, comprehending EAs is fundamental to enhancing EA development and transparency and, thus, bringing a plethora of benefits to every constituent of the EA community. This work addresses the challenges of comprehending EAs and addressing the previously set out research questions by introducing the Population Dynamics Plot (PopDP) to visualise the entire population of a dual- and a many-objective EA population as they evolve during an EA run. We aim to develop a highly informative visualisation that can show an abundance of information in a single plot without overwhelming the user.

In the previous chapter, we were able to create a visualisation that shows the solutions in the objective space or a lower dimension projection of the objective space (efficacious for visualising many objective solutions), allowing one to visualise the journey of solutions. We also provide an analysis of this method to consider the visualisation as an informative tool to visualise the search space. To develop an explainability tool, we use this as a subsection of the methodology to provide the first sector of informativeness for PopDP; we then aim to provide additional information visualising the lineage and the perturbation operator used to create the solution, creating a visualisation that as a package can exhibit the dynamics of the search process and provide a level of explainability about the search process revealing the inner working of the black-box EA. Noting the limitations of possible excessive information, particularly when scaled to larger searches, we provide a framework for which visualising information superfluous to the Decision Maker (DM) can be expunged subject to the DM's requirements. Thus, this provides an easily adaptable and scalable visualisation to fit different problem types, i.e., discrete, continuous, multi- or many-objective problems, choice of visualising solution linage and the perturbation operator generating the solution, and also supports DM preference and accessibility.

After defining the methodology, this work starts by utilising two- and four-objective knapsack problems to demonstrate how PopDP can illustrate algorithm performance, complemented by implicit visual analysis. We then consider visualising the knapsack problems setting a range of different hyper-parameter choices to show the effect hyper-parameters have on the search and demonstrate how EA practitioners can use PopDP as an informative way of visualising the search. Next, we apply PopDP to simpler problems to first create a basis for understanding the visualisation before applying it to more complex problems with a larger search space. Finally, we apply the PopDP tool to real-world offshore wind farm optimisation problems and then analyse and evaluate PopDP considering the XAI benchmarks.

4.1.1 Visualisation Generation

We first aim to visualise the population in the objective space or as close to the objective space representation as possible. If we do this, we can determine areas of the objective space explored and not explored by the search. Furthermore, we can observe artefacts of the problem and EA. We want to preserve the structure of the population to see the journey of the search process. To do this for the two-objective problems, we can simply project the two dimensions into the x and y axes and project some aspect of time in the z-axis, to have a temporal representation of the search. For these visualisations,

we project the generation number on the z-axis. To use a similar methodology for the multi- and many-objective problems, one needs to implement a dimension reduction tool, and so we use the methodology outlined in Section 3.5, which utilises a fast MDS approximator called LMDS to project the solutions to a lower two dimensions which can be plotted in the x and y axes. Chapter 3 provides more details and analysis of this method.

Next, we aim to visualise the reproductive operators which operated on the solutions and indicate the generation of operation. Doing so allows one to determine how the solutions are being generated and how the interaction between reproductive operators affects the search. For example, is there overuse of an operator, such as mutation, causing difficulties converging? Or is the operator enhancing convergence? To capture the reproduction operators which created the solution, we perform this acquisition during the EA evaluation. We operate an EA to solve a problem. During the optimisation process, a record is kept of how each solution was generated and the solution generation mechanism. The record uses an ID for each solution that is generated and, therefore, can be traced throughout the search. When plotting solutions on the visualisation, we colour solutions and forge their shape based on this record. We also use this record to determine solution parent-offspring relationships. With a record of the parent-offspring relationship, we can then implement the final aspect of PopDP by plotting solutions on the visualisation and colouring solutions and forging their shape based on this record. This provides the traceability aspect of PopDP, allowing one to visualise the traceability of solutions through their ancestry tree.

The three aspects of explainability (visualisation of the search space, traceability and the reproduction operator) come together on a single concise plot to produce a population dynamics plot (PopDP). We plot the solutions in their objective space or a dimension-reduced representation and use the parent-offspring and reproduction operator record to colour and forge the solution shape. Any colours and shapes could be used. One could consider accessibility and implement this decision based on the preferences of the audience. i.e., one could use colours or a colour scheme suitable for a visually colour-impaired DM. Alternatively, we could colour based on solutions of interest, such as colouring non-dominated solutions gold. For this work, we colour solutions that are infeasible as red; we assume infeasible solutions are of least importance to the DM, and so we choose not to show the reproductive operator in this instance for infeasible solutions. We colour solutions created by mutations green and forge the shape to a diamond. We colour crossover-produced solutions as black and forge the solution to the shape of a cross. We then use the parent-offspring record and assign solutions produced by crossover with a black dotted line to each parent in the previous generation. Finally, solutions are coloured blue if they have been initially generated or if they are strong solutions carried over from a previous generation.

4.1.2 Experimental Parameters

In this subsection, we list the default set-up parameters; for instances where these differ, we explicitly state them in the figure caption. NSGA-II (Deb, Pratap, Agarwal & Meyarivan 2002) is used for evolving and generating the solutions for a problem. NSGA-II has been set up with the one-point crossover operator and with a probability parameter of 0.3. The mutation operator utilised is the bit flip with a probability set at

0.3. The population size is set to 10 solutions, and the number of generations is set to 10. The algorithm and hyper-parameters have not been optimised. We do not intend to construct an optimised algorithm for these problems as this is not the aim of this work. The artefacts and abnormalities of a non-optimal algorithm can be informative to visualise to the reader. However, this by no means detracts from this work.

The test problem used is the binary multi-objective knapsack problem, which is a discrete problem. The conception of the knapsack problem is to fill a knapsack (or a bag) with objects to maximise gain and not exceed some constraint. This is a common representation of a logistical type of problem. The binary multi-objective knapsack problem was first introduced in the work of Zitzler & Thiele (1999). The dual-, multi- and many-objective binary knapsack problem can formally be defined as: given a set of m items and a set of n knapsacks, where c_i is the capacity of knapsack i, $g_{i,j}$ is the gain (i.e., profit, customer preference) and $w_{i,j}$ is the weight of item j according to knapsack i, the task is to find a vector $x = (x_1, \ldots, x_m) \in \{0,1\}^m$, such that

$$f(x) = (f_1(x), \ldots, f_n(x)) \tag{4.1}$$

is maximised, where,

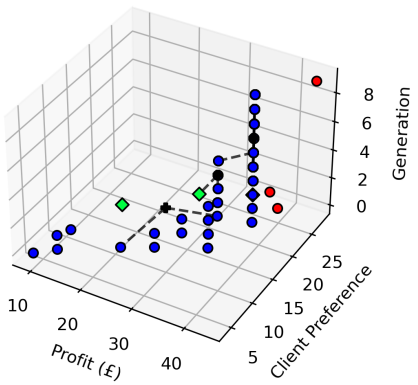
$$f_i(x) = \sum_{j=1}^{m} g_{i,j} \cdot x_j \text{ for all } i \in 1, \ldots, n, \tag{4.2}$$

subject to $\sum_{j=1}^{m} w_{i,j} \cdot x_j \leq c_i$.

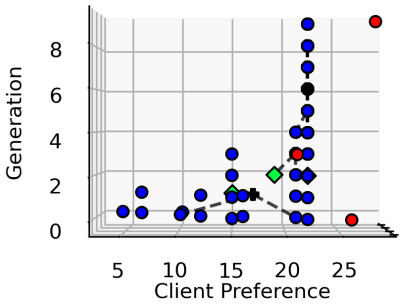
The number of decision variables (possible items in the knapsack) for all objectives has been set arbitrarily to seven. Therefore each chromosome (possible solution of the arrangement of the knapsack) contains seven alleles. The problems contain one knapsack. The knapsack problems contain either two (profits and client preference) or four objectives (profit, client preference, reduction in CO_2 emissions and time between packing score). The problem is dual-objective in Subsection 4.2 and Section 4.3. The problem is four-objective in Section 4.4. The number and type of objectives can be scaled arbitrarily. All four objectives are to be maximised. The single knapsack constraint is a weight capacity of $c_1 = 29$. The maximum objective values for the dual knapsack problems are: 37 profit ($f_1(x) = 37$) and 22 preference ($f_2(x) = 22$). The maximum objective values for the simpler many objective knapsack problems are $f_1(x) = 37$, $f_2(x) = 22$, $f_3(x) = 30$ and $f_4(x) = 19$ and for the complex many objective problem: $f_1(x) = 95$, $f_2(x) = 95$, $f_3(x) = 83$ and $f_4(x) = 55$ with a capacity $c_1 = 99$ and 20 decision variables. We run the complex problem for 100 generations. We have arbitrarily set these parameters of the problem to showcase PopDP.

4.2 Visualising Dual-Objective Knapsack Populations with PopDP

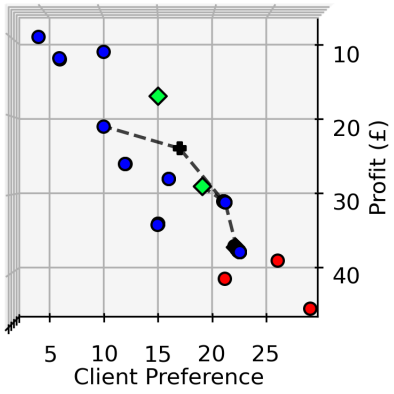
In Figure 4.1 we can observe a PopDP visualisation of the dual-objective results. As described in the methodology, we can see the two objectives, profit and customer preference, in the x and y axes. In the two objectives, we can plot the raw objective data points, unlike in higher dimensions where we would need to plot the dimension-reduced coordinates (usually some combination of the objectives). We see three different visu-



(a) Azimuth and elevation angles are $35°$ and $-60°$ respectively.



(b) Azimuth and elevation angles are $0°$ and $0°$ respectively.



(c) Azimuth and elevation angles are $90°$ and $0°$ respectively.

Figure 4.1: The dual-objective knapsack problem solutions generated with a mutation probability of 0.3 and crossover probability of 0.3. The ◆ solutions were created with mutation. The ● solutions are non-perturbed. The + solutions were created with crossover. The ● solutions are infeasible solutions.

alisations in the figure; this is a single run from the perspective of three different azimuth and elevation combinations to maximise the interpretability for the DM; although the single Figure 4.1a is sufficient, PopDP used interactively can improve interpretability. We can see from the visualisations that solutions created by mutations are represented as green diamonds, and crossover-produced solutions are represented as a black cross with a black dotted line to each parent. Blue solutions represent solutions carried over from a previous generation, and infeasible solutions are represented as red circles.

From the PopDP visualisation, we can observe a number of interesting points. For

example, we can see an infeasible linear decision boundary, and we can observe three of the solutions as being infeasible. From the objective space projection, we can see the objective space visited and not visited by the EA's solutions. This provides one with more information about which areas of the objective space are strong/attracting the EA solutions during the evolution and provide information on which areas of the objective space may require greater exploration. With the generation axis, we can also see the generations of greatest exploration and exploitation and the gradient of change. From the reproductive and parent-offspring indications, one can see where every solution has come from (except the infeasible solutions), how it has been created and from which parents.

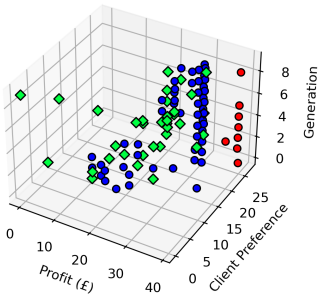
We can see the perturbation effects of crossover and mutation on the search. We can see how crossover has produced a solution midway between two parents, which appears to show crossover to be driving exploitation for this particular instance. Thus, the solution created has not been particularly strong, resulting in the solution doing little to enhance the search at such an early stage. Perhaps crossover could be more effective in the later search for this instance.

4.3 Hyper-Parameter Analysis with PopDP

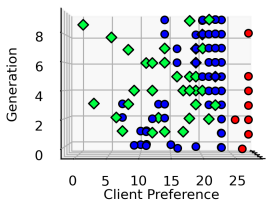
In Figure 4.2, we experiment with three runs and alter the mutation and crossover values for each row. The aim is to determine the effect the parameter values have on the search via a PopDP visualisation to demonstrate PopDP as a tool for analysis. With PopDP, visualisations of the different searches with different values can be compared and analysed. In this demonstration, we consider the extreme values/combinations of the search to magnify the effects. We see in the top row of Figure 4.2 a search with the mutation value set at one and the crossover value set to 0. In the middle row, we show the mutation value set to 0 and the crossover value set to 1, and in the bottom row, we show both crossover and mutation values set to 0.5.

When providing the DM with these three searches, the DM can understand how their solutions were obtained and make adjustments to the search based on the insight obtained by PopDP. The most prominent feature when comparing the visualisations is that we can see the dominant operator in each case. For example, we can clearly see in the top row a visualisation containing many green diamonds, showing mutation to be the dominant operator of that search. Whereas the middle row is entirely dominated by crossover, and the bottom row appears to be a relatively equal mixture of crossover and mutation. We can also observe mutation driving lots of exploration at later stages of EA search as seen by many solutions (created by mutation) with large intersolution distances at the later stages of the search. It appears that for every generation (apart from the initial, where the intersolution distances are equal) of the search, the solution distances are much further apart when compared to other parameter value searches. This is unlike the high-value crossover runs that appear to show greater exploitation/convergence during the search. We see intersolution distances much smaller through the whole search process, and these solutions appear to converge quickly, which may not be optimal for problems with many local optima. For optimisation, neither of these combinations are optimal.

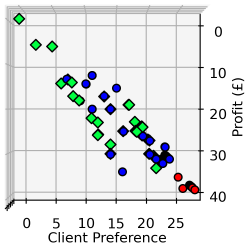
We can also observe from the PopDP visualisation that in some cases (as seen in



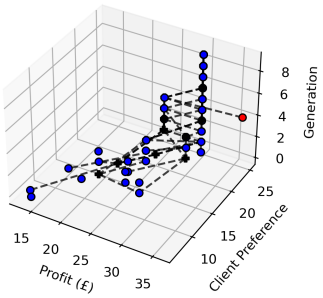
(a) Azimuth and elevation angles are $35°$ and $-60°$ respectively.



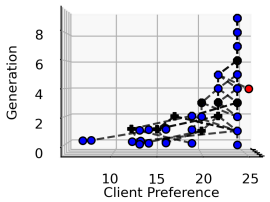
(b) Azimuth and elevation angles are $0°$ and $0°$ respectively.



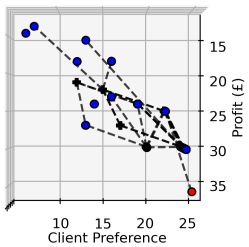
(c) Azimuth and elevation angles are $90°$ and $0°$ respectively.



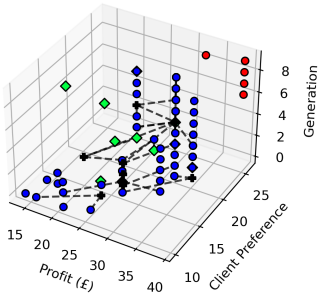
(d) Azimuth and elevation angles are $35°$ and $-60°$ respectively.



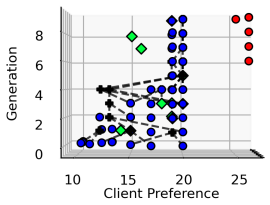
(e) Azimuth and elevation angles are $0°$ and $0°$ respectively.



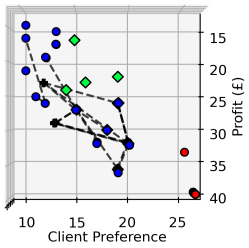
(f) Azimuth and elevation angles are $90°$ and $0°$ respectively.



(g) Azimuth and elevation angles are $35°$ and $-60°$ respectively.



(h) Azimuth and elevation angles are $0°$ and $0°$ respectively.



(i) Azimuth and elevation angles are $90°$ and $0°$ respectively.

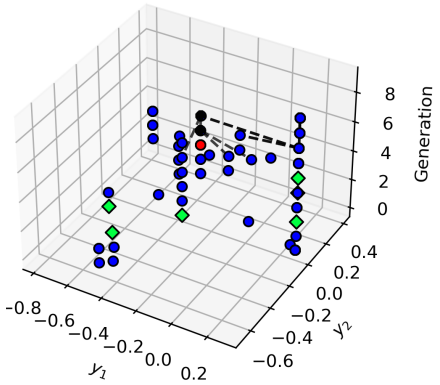
Figure 4.2: The dual-objective knapsack problem solutions. Row 1 has been generated with a mutation probability of 1.0 and crossover probability of 0. Row 2 has been generated with a mutation probability of 0 and crossover probability of 1. Row 3 has been generated with a mutation probability of 0.5 and crossover probability of 0.5. The ◆ solutions were created with mutation. The ● solutions are non-perturbed. The + solutions were created with crossover. The ● solutions are infeasible solutions.

the middle row), infeasible solutions, whilst possibly impractical to the DM themselves, can remain in the search and generate good feasible offspring solutions, enhancing the search. This is perhaps not completely intuitive. We also see high mutation value searches generating more infeasible solutions for this particular problem, or at least the infeasible solution remains in the population for longer because of poor average population fitness - either way, demonstrating this search to be poor and certainly a non-optimal search compared to the other two PopDP plots. One benefit of this search is that it appears the large exploration and lack of convergence does generate a diverse number of solutions to choose from, indicating to the DM that setting a larger mutation value for their search may increase diversity if they wish to compile a larger set of final solutions. Comparing the three PopDP visualisations, we can clearly see that the mixture of parameters in the bottom row is a more effective search than the two previous runs.

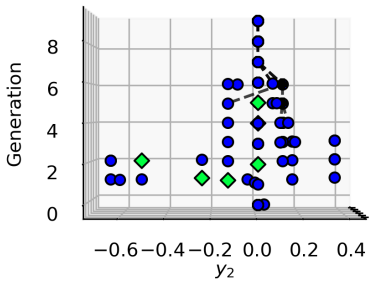
4.4 Visualising Many-Objective Knapsack Populations with PopDP

We now provide a many-objective problem example, and later we demonstrate a more complex problem example. This problem instance is a harder variation of the Knapsack problem used in the dual-objective problem example, requiring a larger runtime (100 generations). In Figure 4.3, we can see the many-objective problem. When contrasted against the dual-objective PopDP, the most prominent difference can be seen in the x and y axes. We can see the reduced dimensions instead of the raw objective output in the x and y axes; these are linear combinations of the four objectives. One can also observe the solutions converging along multiple non-axis trajectories (the multiple objectives). This is unlike the dual-objective problems where the solutions converge along the dual objective axes, forming a single trajectory of solutions; this is an artefact of the algorithm seen in the two-objective problems. We can also observe the infeasible solutions in the middle of the figure, and hence no clear linear boundary exists in the PopDP visualisation. This is due to the amalgamation of objective combinations during the dimension reduction process. However, one can still observe the clusters of strong solutions and the reproductive operators that generated these solutions. One can see the strongest solutions are those solutions in the centre of the figure, with paths of the search trailing toward the centre.

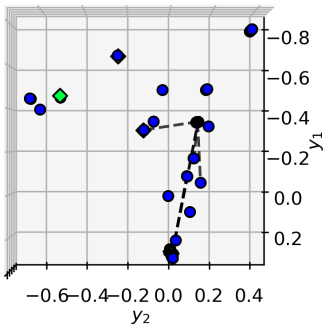
In Figure 4.4, we provide a more complex problem with a greater number of generations to consider how PopDP scales to larger problems. Generally, we can see the parameter choice appears effective for this problem and from a debugging perspective no obvious errors from the search process. However, there is a focus on exploitation slightly too early, as can be seen by the solution exploration up to generation 20 (large perturbations around the search space) and exploitation post generation 20. One can observe that some of the near-optimum solutions have been created with crossover from parent solutions located at the far ends of the explored search space. This means solutions that are strong in a single objective can be crossed to create offspring solutions strong in multiple objectives. Like the previous examples, we can see where the strong solutions tend to cluster. Moreover, we see the key search space areas of convergence, and we can follow the lineage back through time to see the evolution process of these dominant solutions.



(a) Azimuth and elevation angles are 35° and -60° respectively.



(b) Azimuth and elevation angles are 0° and 0° respectively.

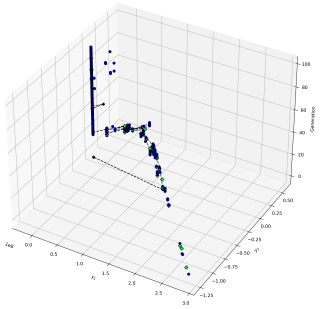


(c) Azimuth and elevation angles are 90° and 0° respectively.

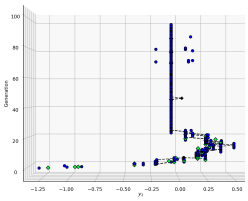
Figure 4.3: The many-objective knapsack problem solutions generated with a mutation probability of 0.3 and crossover probability of 0.3. The ◆ solutions were created with mutation. The ● solutions are non-perturbed. The + solutions were created with crossover. The ● solutions are infeasible solutions.

4.5 Explainable Optimisation for Offshore Wind Farms

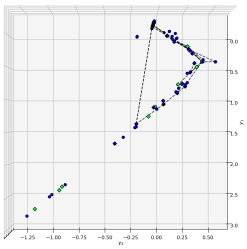
In this section, we consider the application of PopDP to a real-world problem. One key advantage to providing explainability methods is to create an understanding for real-world problems where the results may have major consequences on social, ethical, and financial impacts. Explainability in EC is important for a number of reasons, including debugging and developing EAs, motivating individual decisions supported by AI, and for providing an audit trail. The processes of optimisation must be trustworthy and transparent, which is achieved partially through XAI. We aim to illustrate, using PopDP,



(a) Azimuth and elevation angles are $35°$ and $-60°$ respectively.



(b) Azimuth and elevation angles are $0°$ and $0°$ respectively.



(c) Azimuth and elevation angles are $90°$ and $0°$ respectively.

Figure 4.4: The many-objective 'complex' knapsack problem solutions generated with a mutation probability of 0.2 and crossover probability of 0.6. The ◆ solutions were created with mutation. The ● solutions are non-perturbed. The + solutions were created with crossover. The ● solutions are infeasible solutions.

an explainable optimisation approach applied to a real-world offshore wind farm layout problem.

Due to the effects of climate change and the instability of geopolitics on energy security, many western countries are now prioritising the transition from fossil fuels to renewable energy sources such as solar, hydro and wind alternatives (Eriksen & Hauri 2022). Moreover, increasing investment in renewable energy permits engineers to optimise the extraction of energy. The positioning of wind turbines is affected by wake and other factors which influence the cost of energy production (Mosetti et al. 1994). We shall be optimising and applying PopDP to a problem that considers the output given the positioning of offshore wind farms.

Offshore Wind Farms (OWF) are a collection of wind turbines located in a body of water. The large blades of the wind turbines are connected to generators in order to transform kinetic energy into electricity. In contrast to onshore wind farms, they benefit from the high winds of the sea and minimise visual and acoustic pollution. The positions of the wind turbines should be optimised to maximise the output. The work of Mosetti et al. (1994) developed and optimised, with a genetic algorithm, a wind turbine model considering wake superposition to minimise the cost to power output ratio; they considered three different wind scenario benchmark problems. In this chapter, we apply PopDP to the benchmark problems in Mosetti et al. (1994).

4.5.1 Experimental Configuration

In this subsection, we detail the configuration of the offshore wind farm problem and EA used in this section. The offshore wind farm problem is discrete, with the variables representing the position of a wind turbine in a 2-D grid with 100 possible locations as demonstrated in Figure 4.5. We assume there to be only a single type of wind turbine, a

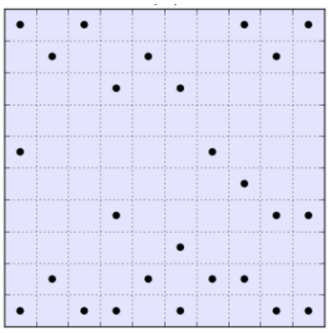


Figure 4.5: The optimal wind farm layout from the work of Mosetti et al. (1994). Black dots represent the locations of the wind turbines.

single wind direction (north to south at 12 m/s), and the cost of the wind farm depends only on the number of wind turbines. The power calculation uses a constant thrust coefficient. The two objectives of this work are to maximise power output (kW) whilst minimising the cost of the wind farm (dimensionless, the number of wind turbines). The wake model implemented was the Katic-Jensen wake model Mosetti et al. (1994), which can be formulated as

$$v = v_0 \left[1 - \frac{\left(1 - \sqrt{1 - C_t}\right)}{\left(1 + \frac{kx}{r}\right)^2} \right], \tag{4.3}$$

where v is the velocity of the wind behind the turbine rotor, v_0 is the initial velocity, x is the distance between turbines, and the thrust coefficient is C_t. Then the wake region behind the rotor is calculated from

$$r = r_0 \sqrt{\frac{1 - \frac{1}{2}\left(1 - \sqrt{1 - C_t}\right)}{1 - \left(1 - \sqrt{1 - C_t}\right)}}, \tag{4.4}$$

where r_0 is the radius of the rotor. With v, one can then calculate the first objective which is the total power output of the wind farm from

$$P = \sum_{i=1}^{N} \frac{1}{2} \rho A C_p v_i^3 \approx \sum_{i=1}^{N} 0.3 v_i^3. \tag{4.5}$$

The second objective is the cost which can be formulated as

$$cost = N \left(\frac{2}{3} + \frac{1}{3} e^{-0.0017N^2} \right), \tag{4.6}$$

where N is the number of wind turbines. The constants for the equations can be found in the work of Mosetti et al. (1994), which has been summarised in Table 4.1.

To optimise the problem, we used a genetic algorithm with Pareto sorting and ranked selection to generate solutions with either binary crossover (10%) or bit flip mutation (90%). At each generation, we combine the parent and child solutions and keep the

Parameter	Value
Thrust coefficient (C_t)	0.88
Rotor radius (r_0)	20
Rotor diameter (D)	40
Wake decay factor (k)	0.0943695829
Air density (ρ)	1.225
Power coefficient (C_p)	0.39
Swept area of rotor (A)	1256.64

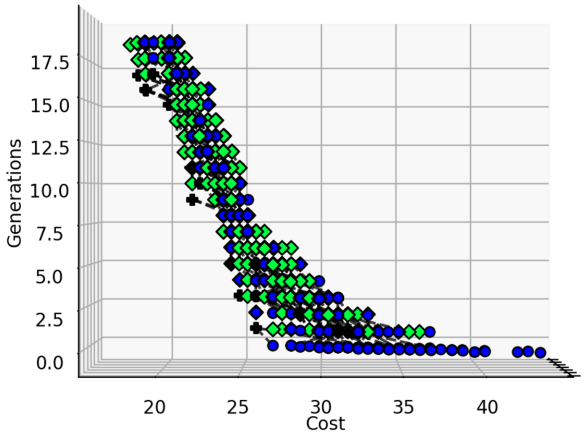
Table 4.1: The objective function parameters from the work of Mosetti et al. (1994)

best 50% as parents in the next generation. We stop the evaluation after a fixed budget of function evaluations. In Figure 4.6 we evaluated for 20 generations, whilst in Figure 4.7 we evaluated for 150 generations. The population size is set to 100 in Figure 4.6 and 10 in Figure 4.7.

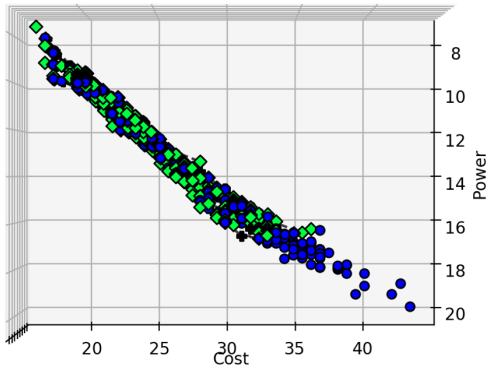
4.5.2 Results

In Figure 4.6, we present the PopDP visualisation of a run with a population size of 100, which was evolved for 20 generations. We can see when the solutions have been generated, by which operators and their corresponding parents. We can also see the journey of the population to the Pareto front and hence the direction of the search. From a much larger quantity of green diamonds than black crosses, we can see mutation as the dominant operator, as we would expect from setting the EA's hyper-parameters (mutation probability set to 0.9 whilst crossover probability set to 0.1). We can see, however, the black crosses showing the ineffectiveness of the crossover operation for this configuration of problem. The operator does not create solutions which remain in the subsequent generation. This shows crossover to enhance exploitation at too early a stage of the search process. We can determine an approximation for the exploration and exploitation from the width of solution coverage at each generation. We can see this as time increases; the width of solution spread appears to decrease, indicating the search yields a more exploitive state over time. Finally, we can also see the traversed search space and the untraversed areas of the search space.

In Figure 4.7, we create a PopDP visualisation of an EA that has been evaluated with a population size of 10 for 150 generations. This problem has been evaluated for more generations than the previous figure. From this figure, we can determine the generation to stop evaluating, as can be seen at generation 120, where the search begins to exploit a single area for a prolonged period. In the case of this benchmark problem, the algorithm converges in 100-150 generations, as shown by the single column of solutions. From the visualisation, we can identify where the solutions begin to converge with a good ratio of exploration in the early stages of the search, followed by late exploitation - indicating a strong run for these hyper-parameters and this instance. Suppose one was to observe exploitation of a single position commencing at generation 20. In that case, we could observe a poor exploration ratio and suggest updating parameters to increase the initial exploration stage of the search. As deep exploitation does not appear to take effect until the final 10% of the search, we deem these hyper-parameters appropriate. Again, like the previous figure, we can see the key search



(a) Azimuth and elevation angles are 0° and 0° respectively.



(b) Azimuth and elevation angles are 90° and 0° respectively.

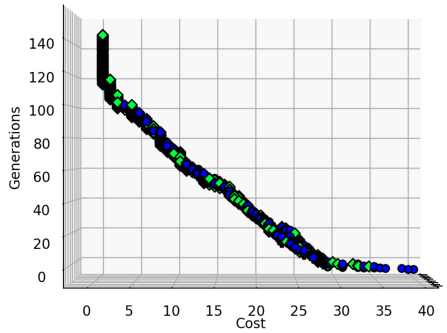
Figure 4.6: The wind farm problem solutions generated with a mutation probability of 0.9 and crossover probability of 0.1. The ◆ solutions were created with mutation. The ● solutions are non-perturbed. The $+$ solutions were created with crossover.

space areas of convergence, and we can follow the lineage back through time to see the evolution process. Ultimately, PopDP allows us to peel back the cloak from a GA to see the inner workings via lineage, solution structure and reproduction mechanisms.

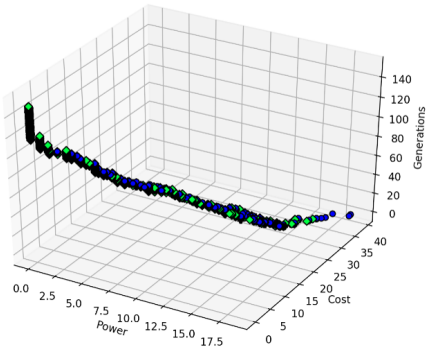
4.6 Discussion on ECXAI

As we have seen from the previous two subsections, PopDP provides a wealth of information about the evolutionary process. Some of the most useful information derived from the visualisations include determining what operators are proving useful and how they affect the search. We are also provided with information such as whether the optimiser has converged and where are the strong objective space areas of the search.

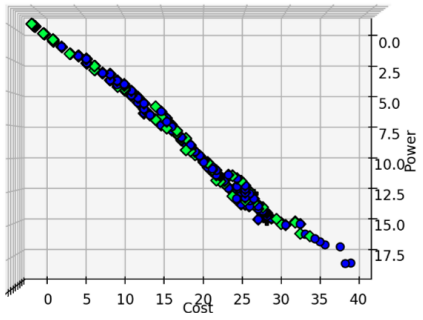
PopDP is an early attempt to produce explainable visualisations of multi-objective op-



(a) Azimuth and elevation angles are 0° and 0° respectively.



(b) Azimuth and elevation angles are 35° and -60° respectively.



(c) Azimuth and elevation angles are 90° and 0° respectively.

Figure 4.7: The wind farm problem solutions generated with a mutation probability of 0.9 and crossover probability of 0.1. The ◆ solutions were created with mutation. The ● solutions are non-perturbed. The $+$ solutions were created with crossover.

timisers. This is the first time it has been applied to a benchmark representing a real-world engineering problem. Considering the real-world wind farm problem, PopDP provides a degree of explainability in numerous ways. First, it provides information about which of the evolutionary operators being utilised are useful and provides, to some degree, the effect the operator is having on the search, allowing for comparability to other results. In the previous evaluation, we can see from the visualisations that most of the successful solutions have been produced using mutation. That is to be expected, and one of the reasons the optimiser was set up in this way is to ensure that known artefacts in the data are observable, but it is interesting to see that later on in the optimisation process, there is still the occasional solution being generated using the crossover operation, and this appears to be supporting exploitation which is needed during the late stages of the search. Secondly, it can provide an indication that the algorithm has converged. That is critical if we are paying for the computing time or resources are limited to perform the optimisation. There is the question of whether this is a technical visualisation that someone with a deep background in optimisation

would understand or whether it is an accessible explanation of a complex process. We feel it is between the two, as a high-level (basic) understanding of the EA process and visualisation is required for interpretability.

As well as describing PopDP in terms of principles used within evolutionary computation and what we intuitively expect a DM requires, it is also important to consider the method through the lens of XAI; therefore, we contrast this work with the nascent literature of XAI to benchmark the degree of explainability produced by PopDP. Taxonomies such as Guidotti et al. (2018) and others commonly classify models into global, local or introspective methods; given PopDP's ability to understand solutions at the individual level, we could classify PopDP as a local method; however, as we have demonstrated, considering the whole process, brings the ability of comparison and so it is more informative to classify PopDP as a global method, given its ability to interpret the EA process at a holistic level. This is the case for other XAI techniques, such as SHAP Lundberg & Lee (2017), which can consider the contribution to prediction for individual solutions, or one can consider the overall model with the amalgamation of solutions. We also note that PopDP is also a model-specific technique, given its ability to only work on specific models (GAs).

As recognised in the existing literature (Phillips et al. 2020), not all AI requires a level of explainability. However, the primary purpose of designing PopDP was to support and comprehend algorithm development and enhancement, so explainability is necessary for all these cases. We also note that there is no universally applicable explanation of explainability in XAI, so it is important to consider the use case of PopDP and its probable user when analysing its explainability (Arrieta et al. 2020). Given PopDP's use case, we might anticipate that its most probable users would either be experts developing or improving EAs or non-experts who have a basic comprehension of the EA framework (i.e., an understanding of crossover and mutation). Now that we are cognizant of the requirements and competencies of the users, we can contrast PopDP against the existing XAI literature benchmarks.

From the existing literature (Phillips et al. 2020), XAI can be divided into four principles: explanation, meaningful explanation, explanation accuracy and knowledge limits. We now evaluate PopDP against these principles to determine a degree of explainability.

First, we consider the explanation, which is the evidence or reason(s) for the output decision. For an EA, the output would be the final solutions, and from PopDP, the explanation for these solutions and the way they were generated would be obtainable from the PopDP visualisation showing the objective space, lineage and reproduction operators. For example, one can see how a solution has been created and from which of the parents it evolved. We also may have an indication of why those parents evolved, such as because the algorithm has a good selection mechanism choosing strong parents to reproduce. Therefore the condition of explanation would be upheld, given PopDP's ability to provide a visual explanation of the solutions. The remaining three principles are properties of the quality of the explanation provided by PopDP.

The second principle considers whether the intended recipient can comprehend the system's explanation(s) and is known as the meaningfulness principle. From the information displayed in the PopDP visualisation, it is assumed the user has a basic

knowledge of the EA process and can follow the evolutionary algorithm search through the generations to observe the evolution of the solutions and the effect operators and hyper-parameters are having on the search. The level of meaningfulness would depend on a number of factors, including the recipient's experience of EAs, PopDP and their cognitive ability. Therefore, PopDP can be assumed a suitable explainability tool, given most people employing evolutionary algorithms to solve a problem have some level of technical competence and a basic comprehension of EAs. In future, a usability study could be undertaken to support these results.

As often, when explaining complex events, simplicity can be increased with an inversely proportional decrease in the fidelity of the explanation accuracy. The explanation accuracy concerns itself with the accuracy of PopDP's ability to display the decision-making process or an EA's inner workings. From the dual-objective PopDP visualisations, one can observe a direct (no-accuracy reduction in the solution position) projection of the lineage of solutions, the fitness values of the solutions and the operators used to create them. However, for the case of multi- and many-objective solutions, one would have to use a dimension reduction technique to reduce the dimensionality of the solution at the cost of reducing the accuracy of the information about the position of the solution in the high dimensional space. Different dimensionality reduction techniques preserve the data in different ways. For PopDP, we suggest using LMDS, shown in Chapter 3 to be an accurate and practical dimension reduction tool for EA populations. Therefore the accuracy of the explanation for the dual-parameter case is high as it directly represents the relevant aspects of the solution, whilst the visualisation of the multi- and many-objective results conserved most of the information and yielded a high accuracy but at a slightly reduced cost of accuracy during the dimension reduction process. As the majority of information is captured in both cases, we would say this explanation accuracy threshold is upheld. While we have not observed any issues for the problems evaluated in this chapter, there may be scalability issues which could affect the fidelity of the explanation; for example, by extending the number of function evaluations for the optimiser or the types of problems, it may diminish the ability to glean useful explanations about the evolutionary process (this is the case for many visualisations).

Finally, the work suggests knowledge limits are necessary to ensure the AI is not operating outside of its intended boundaries. Early experimentation has not identified a case where this visualisation is inappropriate, but some extreme cases may exist. This is likely true for most visualisations, given a large enough number of problems and algorithms to test. However, categorising data based on the difficulty of visualisation could narrow the search to finding a proof by counterexample.

We also considered other metrics used for benchmarking explainability from other authors in the existing literature. For example, the works of Kim et al. (2015), Ribeiro et al. (2016), Arrieta et al. (2020) all considered the requirement of trustworthiness of the AI by explainability methods. The use of PopDP to reveal the inner workings of a search could provide a better understanding, greater user confidence and trustworthiness that the EA is doing as intended during the search. Furthermore, the work of Arrieta et al. (2020) recognises the significance of transferability that XAI methods should provide to the user. In the results of this chapter, PopDP was used to find limitations and weaknesses of an EA during the search process and so could be used to help define

EA boundaries for supporting EA transferability within different problems. The work of (Arrieta et al. 2020) notes that interaction with AI can enhance interpretability. As discussed in this chapter, PopDP has the capability to be interactive and tailored to the users' requirements allowing the recipient of the interactive visualisation to engage with the process in additional dimensions and change the way information is displayed to suit the user. Additional principles can also be found in the literature of Phillips et al. (2020), Arrieta et al. (2020), Guidotti et al. (2018), Confalonieri et al. (2021) and Vilone & Longo (2020). However, many other principles overlap with the principles we discussed, or it seems in some cases, the authors of these principles did not consider the explainability of EC when developing these principles, likely due to the novelty of ECXAI and are therefore not relevant to explainability in EC optimisation (e.g., privacy) therefore we choose to omit these.

There is more to be done in understanding the extent of its accessibility, and an aspect of ongoing future work should be a usability study with real-world ORE engineers to see how they interact with the visualisation. There are also technical extensions to be considered, such as better highlighting the trade-off between the solutions so that the final Pareto optimal approximation is more visible in the later generations; this is important, as that is where the solutions likely to be chosen will be.

For ECXAI in the context of wind farm operational research, we have proposed a visualisation providing a degree of explainability and traceability to an EA via lineage, solution structure and the reproduction mechanism for many applications and demographics. Two limitations were considered for this work. The first possible limitation was scalability. The figure could become crowded for large crossover values and large populations with such a large volume of information in a confined space. One possible way to mitigate this issue is to provide PopDP as an interactive tool, providing a level of tailorability that can support the DM; it could allow the DM to only focus on specific regions or generations of interest during the search, reducing superfluous information subject to the DM's requirements. The interactive format could be used to zoom into regions of interest as well as to zoom out and get a holistic overview of the search; this could potentially mitigate this scalability issue. The second limitation is the metrics for determining the explainability degree of PopDP (and other methods). The XAI literature is starting to propose quantifiable metrics (Hoffman et al. 2018); however, it may be worth re-evaluating PopDP's explainability when the metrics mature when we may be able to quantify the degree of explainability. We could also complement the implicit explainability evaluation metrics with a usability study to determine the degree of explainability.

Explainable AI is an important component of modern AI projects, mostly machine learning, which has been used within limited ORE projects – but should also include optimisation. With PopDP applied and evaluated to a real-world problem, we have demonstrated the benefits and informativeness it can bring to the problem. When considering future work, the adaptation for use in many-objective problems (as done in the first section) could be applied and the problem evaluated for PopDP's scalability. Finally, we should see an application to a wider range of optimisation problems within ORE and the benefits explainability can bring.

4.7 Conclusion

In this work, we have recognised the requirement for explainable methods in EC. Recognising that many existing XAI methods are not applicable to EC and only applicable to machine learning for the reasons stated, we have developed one of the earliest methods to provide explainability. We recognise visualisation is an effective foundation to develop XAI; it is a preferred method within the existing works of XAI, given the natural human ability to recognise visual patterns quickly (Arrieta et al. 2020, Vilone & Longo 2020), making it an optimal tool for creating XAI methods. However, we also need to distinguish a difference between visualisations for ECXAI and conventional EC visualisation, as the two are not equivalent. For a visualisation to be explainable, it must provide a high degree of informativeness about the process from which it was generated. For example, a heatmap of solutions could not be considered explainable as it does not provide the user with any information other than the final solution values. The ECXAI community, along with EA practitioners and researchers, should work together to form a more thorough definition of ECXAI to define what is and is not ECXAI. Although, this may take some time as the definition of XAI, which has been in existence far longer, is still contested today. In this work, we argue PopDP is explainable against a set of questions (found in the motivation) it can or cannot answer. One possible way to try to quantify this explainability would be for the EA community to list a set of rules/questions that should be answered by an explainability method, applying weights based on the importance of the rule and then counting the number of requirements the method satisfies.

Before setting out to create PopDP we considered the end user, the applications and the use case for the method. To evaluate, we demonstrate PopDP's capabilities in providing a degree of explainability to EAs. In future work, other ORE applications and the application of PopDP to continuous problems could be considered. Also, applying PopDP in the parameter space and comparing the mappings between the objective and parameter space PopDP may be additionally informative. More generally, other ECXAI methods could be developed considering the same framework of considering users, problems and requirements. The degree of these methods and of PopDP should then be compared and examined, perhaps with a combination of metrics (yet to be created) and usability studies to determine the advantages/disadvantages of each method and the best applications. We believe it is unlikely that a single method will be optimal for all use cases. We should also consider the requirements and possible effects of ECXAI by collaborating with real-world users of EC to ultimately provide methods which can enhance and further the use of EC in industry.

We should also consider the use case of EA developers and their requirements for methods that assist with the enhancement and development of EAs. Most of the benefits to PopDP will be to EA developers and researchers; we believe we must first start by creating XAI for researchers before creating XAI for the layperson. We should also consider optimising the type and quantity of information displayed to the DM; this could likely be done with the support of EA practitioners, this could prevent overloading of information or cluttering of the visualisation, and the development of an interactive tool could mitigate the overwhelming amount of information. We also need to understand how much knowledge is needed to comprehend the process. We should also consider

ECXAI methods as ways of supporting the communication of algorithms. ECXAI could then one day be used in publications relating to new technology in EC to communicate algorithms and their search processes to other researchers. With better explainability, we can better communicate, develop trust and enhance EC, benefiting all EA stakeholders.

Chapter 5

Intelligent Visualisation: Enhancing Optimisation Visualisations with Deep Learning

This work proposes, develops and examines deep learning predictive models for identifying solutions in optima in an evolutionary optimisation search. These models have many possible applications, from evolutionary algorithm development to landscape analysis. For this work, we employ the predictive models to enhance visualisations of the solution — the automation of locating optima within the visualisation to assemble a potentially informative, intelligent visualisation. The results exhibit the effectiveness of the predictive models' capability to learn and generalise the characteristics manifested by the evolutionary algorithm during a search and thus transfer learning between problems, allowing one to make optima predictions on the solution data of untrained problems.

EVOLUTIONARY algorithms (EAs) utilise biologically-inspired operators to perturb a solution or population of solutions to the optimal objective value(s). The benefits over conventional optimisation make them essential for optimising challenging problems abound in industry and research. Notwithstanding, EAs possess their own inherent challenges. One fundamental problem in modern evolutionary computation is understanding the search landscape induced by the fitness function and the presence of local and global optima in the landscape. Global optima are the regions that a DM aspires to find, a region of solutions with the best fitness values. Whilst local optima are regions of the space which are not globally optimal but with all near neighbouring regions yielding worse fitness values resulting in ineffective small perturbations of solutions to better space and creating the illusion that the optimiser has converged. A lack of landscape understanding creates challenges - for example, does the landscape contain local optima that will make the search more of a challenge? Are the perturbations of the solutions unsuccessful at overcoming the selection pressure because of local optima in the problem landscape? Or have the solutions converged to the optimum/optima? This work seeks to alleviate these issues by proposing deep learning (DL) models that, given a time series of EA data (objectives and parameter values for solutions in an EA population), predict when solutions are located within an optimal state. These optima detection models could be utilised for many kinds of outputs, for example, detecting when solutions are located in optima and automatically updating the EA hyper-parameters to suitable values - building intelligent self-parameter adaptive EAs or mapping a solution landscape with a fast automated local optima detection/colouring. However, this work considers the models utilised for enhancing the informativeness of EA population visualisations, as the visualisation may provide an

intuitive way to introduce these prediction models.

This research works toward the development of novel Intelligent Visualisations (IViz). IViz uses machine learning (ML) models, more specifically the subset of DL models, to enhance and automate visualisations through predictive modelling. In this chapter, we develop optima-predicting models using EA generated solution data for a particular problem. Then, we evaluate these predictive models on EA data from different untrained problems (e.g., we train the model on EA solutions evolved on the Ackley problem and evaluate the model using Rastrigin EA solution data).

Remarkably, the author can detect optima from solutions generated by untrained problems through the DL algorithm's ability to learn the characteristics exhibited by the EA as it evolves solutions through the search space and, hence, transferring learning from one problem to another problem. However, this ability to detect optima using the visualisation is yet to be confirmed more generally with usability studies by the end users. The optima prediction models can be integrated with conventional visualisations to automate the process of identifying points of interest, such as local optima. In this work, the models show promising results for assembling informative visualisations with vast applications from EA development and explainability to fitness landscape analysis. This work considers using ML to enhance EC processes and we note the volume of existing work utilising ML to enhance EC is much smaller than the literature considering EC to enhance ML. Some examples of ML being used to enhance the EC processes include ML surrogate objective modelling (Singh et al. 2022) and the use of ML to support algorithm selection (Kerschke et al. 2019, Smith-Miles 2009). The novel contributions of this chapter are as follows:

1. DL models are developed for predicting solutions that are optimal.
2. The optima prediction models are qualitatively (visually) and quantitatively (statistically) evaluated on dual-parameter single-objective test problems.
3. We demonstrate the enhancement of visualisations with DL (IViz) to automate the process of finding optima in three-, four- and five-parameter test problems.
4. We provide model explainability with an eXplainable AI technique (XAI).

5.1 Intelligent Optima Detection Methodology

5.1.1 Intelligent Visualisation Data Acquisition

EA Specification

A supervised DL model must be trained with model input data and labels. We start by acquiring the input data and labels so that the model can learn and recognise the characteristics of the EAs during the search (this could be a lack of effective solution perturbation when located in optima, among other reasons). The model needs the input data to be informative, and in order to find the relevant patterns and relationships within the data, input data must be representative of the data produced by the EA.

We start by running a one parent, one child elitist evolutionary strategy (a (1+1)–ES) to evolve solutions through a search space and collect the input data. We use a (1+1)–

ES for simplicity, making the methodology and results more explicit. The mutation probability is set to 0.1. The distribution index, controlling the size of the perturbation, is fixed at 7 for polynomial mutation. At each generation, the parameter value/s and the objective value/s are returned and archived into a dataframe. We evaluate a problem with an EA for 100 function evaluations, and for each problem (see Table 5.1 for the list of problems used in this work), we perform this data collection 5000 times with different random initial solutions. This will allow one to create a dataset with dimensions of $D+M$ columns (D parameter values and M objective values) and 100×5000 rows. It is important this data is not shuffled as we aim to input order-dependent time series vectors into the machine learning model.

Once we have a dataset, we need to pass it to the machine learning data processing pipeline to apply the appropriate data manipulations and transforms. If we were to feed the ML model the raw parameter values and objective values, the model could overtrain and focus too much on the positions of solutions and hence the training problem landscape. Therefore, it performs poorly when the same model is applied to testing data from a different problem. To mitigate this generalisation issue, we convert the features into domain-invariant features. To do this, we first convert the parameter values and objective values into the perturbation size distance from the previous value. Therefore we are now considering the solution values change over time rather than the coordinates of the solution. We then need to normalise this data to create a common scale for each feature between a region of $[0,1]$. Sometimes standardisation may be preferred to provide improved model performance where the distribution of parameter and objective perturbation values follows a Gaussian distribution, and we have a large number of outliers.

Label Generation

Once the input data has been obtained as previously described, we need to obtain labels so that the supervised DL model can learn what to predict. In the case of this work, we wish to create a binary optima classifier; therefore, we need to obtain the dichotomous labels of the solutions based on whether the solution is optimal or not optimal. For other ML detection models, one would need to develop labels with the prediction objective in mind.

We obtain the labels for the set of corresponding parameter space perturbation input data $X = \{x_1, x_2, \ldots, x_n\}$ by considering each solution's objective function $f(X) = f(x_1, \ldots, x_n)$ and comparing the solutions objective values with the solution's neighbouring search space. We do this by sampling the neighbouring search space. As all the problems we consider are minimisation problems, for the single objective problems, if a solution has a lower value than all sampled neighbours, then we consider this solution in a local optimum. We consider a solution ($f_i(X)$) in optima for M-objective problems if it dominates all neighbouring samples ($f_j(X)$). More formally:

$$f_i(X) \prec f_j(X) \iff \forall M(f_{iM} \leq f_{jM}) \wedge \exists M(f_{iM} < f_{jM}). \tag{5.1}$$

To obtain the neighbouring search space, one can apply a small perturbation to the solution with some conditions to ensure the sampling is performed in all objectives. For example, for evaluating the single-objective dual-parameter problems, we sample

and evaluate neighbours of the solution by adding a small perturbation value ε to the parameters (representative of the algorithms mutation); this value can be negative, and the value can be zero for unperturbed solutions $f(x_i^*) = f(x_i + \varepsilon, x_j + \varepsilon)$ $(i \neq j)$ (for the dual parameter case) and evaluating the newly created parameters objective values

$$f(X^*) = \bigcup_{i \in \{1,2\}} \{f(x_i^*)\}. \tag{5.2}$$

For a single-objective minimisation problem, if any of the neighbouring objective values have a lower objective value than the considered solution, the solution is not considered optimal (i.e., it is not the case that $f(X) \prec f(X^*)$). Otherwise, the solution is in an optimal state. We extend to sample from a hypercube around the solution for greater than two parameters. For multi-parameter or multi-objective problems, we can extend the equation 5.2 local optima definition.

The perturbation value used is problem dependent and can be found in Table 5.1. This is because the problems exist over different scales, e.g., Schwefel has the range $[-500, 500]$ whilst Ackley has the range $[-5, 5]$. Generally, the smaller the perturbation value, the more accurate the prediction. However, suppose the value is too small. In that case, all solutions will be labelled not in optima since, as in practice in a continuous space, unless the solution yields the exact theoretical parameter values for the optima (which is an unlikely event), there will always be neighbouring space with a lower objective value. Therefore these values have been approximated through experimentation in this work. Other methods could also be available for finding the optima, including the work of Ochoa et al. (2008), which works by recursively running a hill climber to determine optima.

Experimental Functions

The test problems implemented in this work are omnipresent in the evolutionary optimisation literature. For simplicity, we start with test problems that are single objective problems with two parameters. We chose single objective functions with two parameters to keep the work simple when demonstrating the methodology, with a visually manageable number of spatial dimensions (three). Then we extend to three- and five-parameter variants. We developed and tested models from a mix of multi- and unimodal test problems. However, this work will be on the more challenging multi-modal problems. Table 5.1 provides a list of the problems used and their corresponding objective functions.

5.1.2 Recurrent Neural Networks and Long Short Term Memory Networks

The content inputted into a model is important. Even for deep learning models which decide on the importance of the features given, if a feature, which contains an important pattern, is not present in the input data, then it cannot use that feature to make predictions. We aim to develop predictive models which capture the patterns of an EA evolving solutions to optimal states; we hypothesise that these patterns are a function of time and that the patterns exist in sequential data. Some common networks which can be used to make predictions over sequential data are recurrent neural networks (RNN), long short-term memory (LSTM) and convolutional neural networks (CNN). For the best model performance on the dataset, much of the work considers using RNNs

Problem	Objective Function	ε
Ackley	$f(\mathbf{x}) = \begin{array}{l} -20\exp\left(-0.2\sqrt{0.5\left(x_1^2+x_2^2\right)}\right) \\ -\exp\left(0.5\left(\cos(2\pi x_1)+\cos(2\pi x_2)\right)\right)+e+20 \end{array}$	0.5
Cross-in-tray	$f(\mathbf{x}) = -0.0001\left(\left\lvert \sin(x_1)\sin(x_2)\exp\left(\left\lvert 100-\frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right)\right\rvert+1\right)^{0.1}$	0.25
Himmelblau	$f(\mathbf{x}) = \left(x_1^2+x_2-11\right)^2+\left(x_1+x_2^2-7\right)^2$	1.0
Rastrigin	$f(\mathbf{x}) = An+\sum_{i=1}^{n}\left(x_i^2-A\cos(2\pi x_i)\right)$ where $A=10$	0.5
Hölder table	$f(\mathbf{x}) = -\left\lvert \sin(x_1)\cos(x_2)\exp\left(\left\lvert 1-\frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert\right)\right\rvert$	1.0
Styblinski–Tang	$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}\left(x_i^4-16x_i^2+5x_i\right)$	1.0
Bird	$f(\mathbf{x}) = (x_1-x_2)^2+e^{[1-\sin(x_1)]^2}\cos(x_2)+e^{[1-\cos(x_2)]^2}\sin(x_1)$	1.0
Schwefel	$f(\mathbf{x}) = 418.9829d-\sum_{i=1}^{d}x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	40.0

Table 5.1: A list of the eight multi-modal problems employed in this work. The perturbation value ε is provided for each of the problems.

and LSTMs. So we start by describing and motivating the use of RNN and LSTMs.

RNNs, unlike feedforward networks, consider using outputs of the layers to connect back around to reinput into the layer for future predictions, e.g., the RNN cells receive their previous output y_{t-1} and the new input x_t when computing the cell activation value y_t, therefore by using the previous predictions for computing future prediction we create a kind of memory as future inputs are somewhat dependant on past inputs. This can be seen in Figure 5.1 After enrolling the cells through time, backpropagation is applied. Both the input and output are now vectors. For the first-time step y_{t-1}, the value is usually set to 0 by default.

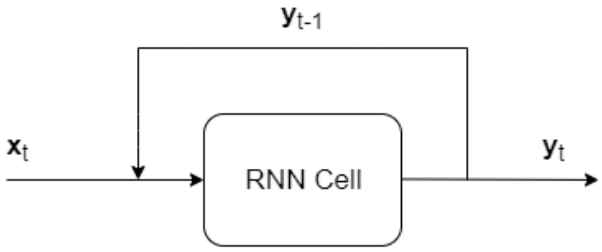


Figure 5.1: Diagram of a single RNN cell.

In this work, we shall consider sequence-to-sequence networks. RNN can be a sequence-

to-sequence network in which a sequence, such as stock prices over the last seven days is inputted into a model and can output the predicted stock price for tomorrow. They can also be a sequence to vector (e.g., sentiment analysis for text) or vector to sequence (used in language translation). RNNs have two main limitations: unstable gradients and limited short-term memory, which can be increased with the use of LSTM networks.

LSTMs can detect longer patterns and thus have a greater memory (Géron 2019). LSTM differ from RNN as they contain a forget gate, an input gate and an output gate with values normalised by the sigmoid function to between $[0,1]$ where a value of zero for the output gate will output everything and a value of one would output everything. These gates are neural networks which choose what information is forgotten or remembered by the cell.

The process of an LSTM network is as follows: we first concatenate the previous hidden state h_{t-1} (hidden output) and the current input x_t. Then we feed this concatenation $[h_{t-1}, x_t]$ into the forget layer to 'forget'/remove the non-relevant information, this output is rescaled with the sigmoid function to a value between $[0,1]$, where a value of 0 will forget everything, we denote this output (f_t). We then update the cell state c_t (which contains memory) by the input gate - the concatenation is fed into the input layer (u_t) - this layer decides what data from the candidate layer should be added/updated to the new cell state and is passed through the sigmoid function. We also push the concatenation through the tanh function to regulate the network value between $[-1,1]$. The combination of the tanh output and the sigmoid output from the previous two steps are multiplied (the sigmoid decides what is kept from the tanh output). Next, we calculate the cell state; the previous cell state c_{t-1} gets multiplied by the forget vector we first calculated (f_t). Then we multiply this output by the output in the input gate to create (c_t). Finally, the output gate (o_t) which decides what the next hidden state should be, is calculated by passing the concatenation $[h_{t-1}, x_t]$ by the new state (c_t) after applying sigmoid and tanh transformations.

5.1.3 Network Development

The previous section describes the methodology for obtaining the model inputs and labels representative of an EA evaluation of a problem, a prerequisite for training. With this multi-variate time series data, we now endeavour to train a model to classify whether a solution lies in an optimum during a search.

Network Architecture

For selecting a model architecture, we decided to experiment with a number of possible models with different levels of complexity and evaluate which model performed best. We considered recurrent neural networks (RNNs) (Rumelhart et al. 1986), long-short term memory networks (LSTM networks) (Hochreiter & Schmidhuber 1997) and convolution neural networks (CNNs) (LeCun et al. 1999). All these networks have been chosen based on their ability to handle sequential data to capture the temporal effects of time series data. These networks are commonly used in time series applications (such as for audio, text, media inputs and so on) to make predictions based on a series of actions at multiple time steps. As we want the model to learn the characteristics of an EA as it evolves solutions over time, these models are most appropriate with

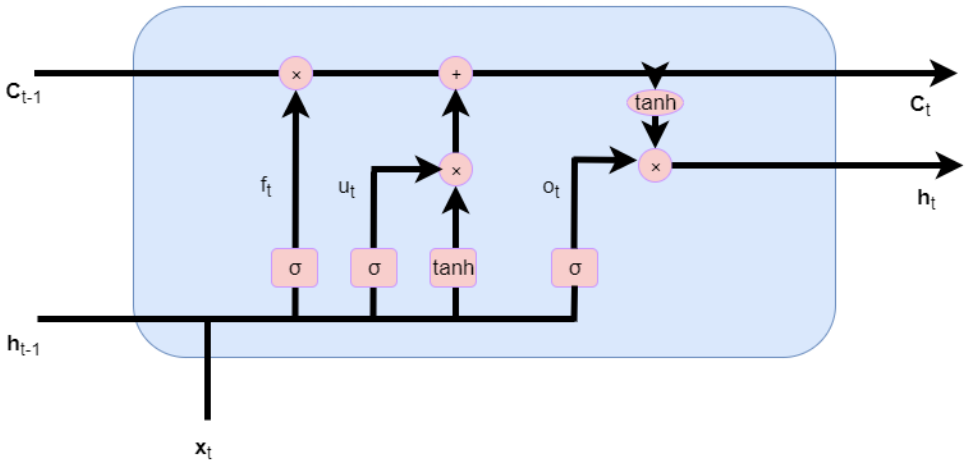


Figure 5.2: A diagram of the LSTM cell.

their ability to learn over time steps using the memory ability produced with RNNs, CNNs and LSTMs. For small sequences of time series data, RNNs are appropriate. However, RNNs have disadvantages such as potentially unstable gradients and limited short term memory (approximately adequate up to 10 time steps (Géron 2019)), which would potentially ignore longer patterns. LSTMs usually have a greater memory capacity allowing them to remember up to 100 time steps (Géron 2019). Furthermore, the sequence to vector CNN has been proven to be effective at learning both short and long term patterns, such as the WaveNet model (Van Den Oord et al. 2016).

Both RNN and LSTM networks produced the best results from the models we tested. However, the most effective model was a sequence to vector RNN. This indicated that for these test examples, perhaps only simple or short term patterns were detected and learnt by the model. Extending this work to more complex algorithms may require the use of more complex model architecture, like LSTMs. See Figure 5.3 for more details on the model architecture, such as model architecture hyper-parameters.

We use the Tanh activation function to mitigate disappearing gradients for all but the final layer; the final layer uses the sigmoid function to produce a value between $[0, 1]$ which, combined with a binary output mapping, is the required output for this dual-classification model. The batch size was set at 32 (a common default parameter), the number of epochs was set to 30 for dual-parameter problems and 60 for all other parameter problems, and the learning rate was set to 10^{-5}. Most of these hyper-parameter values have been selected based on simple experimentation or well-known default parameters, and although this is sufficient for demonstrating the concepts of this work, it is likely hyper-parameter optimisation (e.g., via grid search or Bayesian optimisation) would produce more optimal results. We also experimented with batch and layer normalisation, which can smooth the loss function and hence allow one to use larger learning rates to improve the efficiency of learning, but as most of the model

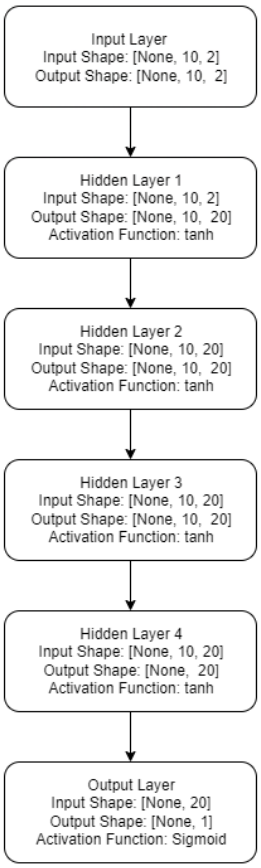


Figure 5.3: The model's RNN network architecture.

training was performed in less than an hour on a desktop computer, we omit these layers for simplicity.

5.1.4 Training the Network

Once a model or series of models has been chosen, we need to preprocess the data to ensure it complies with the model input requirements and that the data is in the best state for training. We first process the data by normalising the dataset. Next, we split the testing and training data. We train on the training problem data and validate on an unseen sample (20%) of the training problem data to try to capture the more general characteristics of the evolutionary search. Binary cross-entropy is used to calculate the loss. Finally, we balance the training data to improve the performance of the minority class. Balancing the data can be done by applying class weights to the training. For an abundance of data, one could restrict the majority class to the size of the minority class, although this would reduce the amount of usable data, so may only be appropriate when limitations of dataset size are irrelevant. Other techniques can be utilised for preprocessing unbalanced data, such as using synthetic sampling of the data, namely,

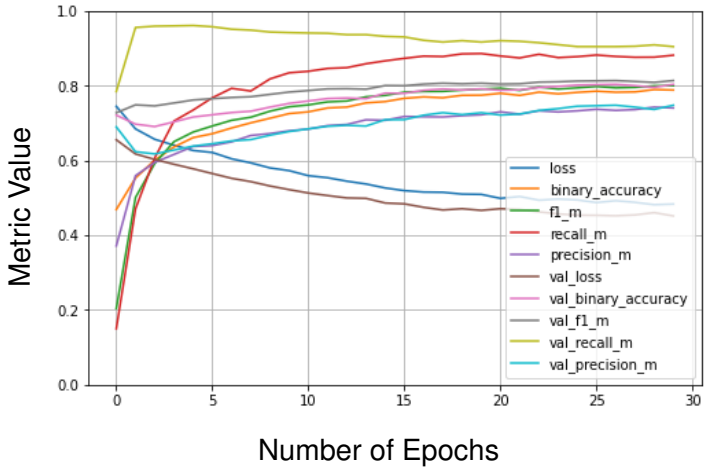


Figure 5.4: Training a model using Bird data. The x-axis displays time, and the $y-$axis displays the metric value.

SMOTE (Chawla et al. 2002), and many of its flavours, alternatively ADASYN: Adaptive synthetic sampling approach for imbalanced learning (He et al. 2008). As the test problem datasets sometimes contain unbalanced classes, the accuracy metric is not an entirely reliable measure of the algorithm's performance. Instead, we utilise precision, recall and the F_1 score (the harmonic mean of precision and recall) to grade each model. The F_1 score can provide a model with a value of between $[0, 1]$ where 1 is the best possible value. The F_1 score can be formulated as

$$F_1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}, \tag{5.3}$$

where

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{5.4}$$

and

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}. \tag{5.5}$$

For the EA evaluation, we used the Platypus library in Python. In addition, we used Tensorflow, Keras, and Scikit-learn for the DL training. The runtime for training lasted no more than one hour for each problem with an Intel Core i7-11700K CPU (3.60GHz), 32GBs of RAM and an Nvidia RTX3080 10GB GPU.

Figure 5.4 shows the training of an optima prediction model for the Bird test problem. See the legend for the different evaluation metrics. We see loss metrics values decreasing and the accuracy measures increasing during the training process. To conclude this section, we provide an overview of the methodology in Figure 5.5.

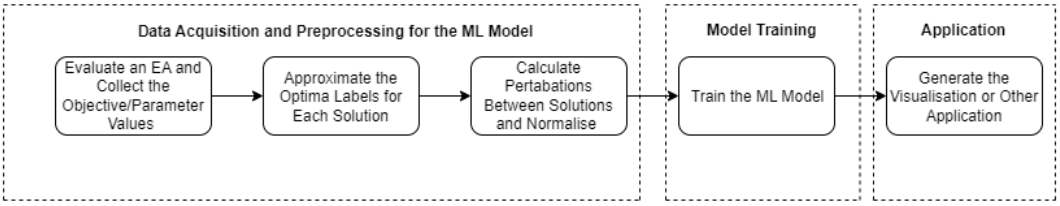


Figure 5.5: An overview of the predictive model methodology.

5.1.5 Visualising Optima via Deep Learning

Creating the Visualisation

After generating the optima prediction models, we can create the visualisation. For the case of identifying optima for dual-parameter problems, we visualise the solutions in a scatter graph with the x and y axes representing the parameter values of the solutions and the z-axis representing the objective value of the solutions. In the case of visualising multi- or many-objective problems, one would need to use a visualisation method that utilises dimension reduction. The work of Chapter 3 uses Landmark Multi-Dimensional Scaling to produce visualisations of the search process with a fast MDS approximator method that has been demonstrated to work accurately for common benchmark problems, such as the DTLZ and WFG suite, whilst simultaneously the MDS approximator allows it to scale well. The method works by projecting the objective and parameter values into a two-dimensional space to plot in the x and y axes whilst presenting time in the z-axis. More formally: given a pairwise $n \times n$ distance dissimilarity matrix of landmarks (arbitrarily chosen) $\mathbf{D} = (d_{i,j})$, find n vectors $x_1, \ldots, x_n \in \mathbb{R}^K$ such that $||x_i - x_j|| \approx d_{i,j}$ for all $i, j \in \{1, \ldots, n\}$. With the created eigenvector $\mathbf{E}_k$ and eigenvalue Λ_k matrices, the $k \times n$ matrix $\mathbf{Y}$ can be constructed by

$$\mathbf{Y} = \mathbf{E}_k \Lambda_k^{\frac{1}{2}}. \tag{5.6}$$

Then, we use distance-based triangulation to plot the remaining non-landmark points. Finally, plot all points in the x and y axes and the generation number in the z- axis.

To use the 'intelligence' and automate the process of predicting points of interest (in this case optima), we use the DL trained model to pass in unseen (data from untrained problems) problem input values and generate a binary output value for each solution. In these models, if the output is the value 1, the solution is predicted to be in optima and is thus coloured yellow. Otherwise, it is not in optima and so is coloured purple.

Visualisation Encoder

For most problems, we use an encoder to smooth the optima approximation colouring. Because this is a statistically based method for approximating optima, we acknowledge the presence of errors in the results. However, fortunately, the true positives and neg-

atives far outweigh the errors (false positives and false negatives). For use within the visualisation of large population sizes, these errors are noticeable and stand out even though it is clear from the visualisation where the optima are located or the clusters of solutions trapped in optima. To make the detection of optima more conspicuous, we provide an encoder which smoothes the solution colouring. For other applications of these models, this encoder may be unnecessary or optional. The encoder works by considering each solution's prediction (either 0 or 1) and its κ nearest neighbours predictions and calculates an average. For solutions returning an average value of greater than 0.5 we label the solution as in optima, and for anything else, we label the solution as not in optima. We provide psuedocode for this process in Algorithm 2.

Algorithm 2 Visualisation Smoothing Encoder Psuedocode

Require: $pop \longleftarrow$ Population of solutions ▷ $pop(i, j)$ where i is the solution number and j parameter values

Require: $l \in \{0, 1\} \longleftarrow$ Labels for the population of solutions

$l_{smoothed} \longleftarrow ()$ ▷ Initialise empty array
for sol in pop **do**
  $\Delta \longleftarrow dist(sol, pop)$ ▷ Euclidean distance array (remove distance to self)
  $\Delta_{sorted} \longleftarrow sort(\Delta)$ ▷ Sort in ascending order
  $N \longleftarrow \Delta_{sorted}(: \kappa, :)$
  $\mu \leftarrow mean(l(N))$ ▷ Mean value of κ nearest neighbours
  if $\mu > 0.5$ **then**
    $l_{smoothed} \longleftarrow 1$
  else
    $l_{smoothed} \longleftarrow 0$
return $l_{smoothed}$

5.2 Dual-Parameter Intelligent Optima Detection

This section provides the results and analysis. This work considers training the model on EA data evolved from a problem and tests the model's ability to predict optima on another untrained problem. As the primary objective is to produce a visualisation, identifying the number of optima and correct locations is the most critical metric for producing an informative optima predicting visualisation. We do, however, consider explicit methods such as the F_1 score as indicators for evaluation.

5.2.1 Single Run Visualisation

In Figure 5.6, we demonstrate a single EA run for 100 generations, generating 100 solutions for this (1+1)–ES. This model has been used to predict unseen/untrained solutions for the Bird problem, with the model returning a testing F_1 score of 0.88. We can see the general shape of the Bird problem even with as few as 100 solutions (this figure can be compared with Figure 5.9, which shows a Bird problem with more solutions). We see all the solutions in the upper half z-axis (higher fitness) of the figure coloured purple to distinguish solutions in non-optima and most of the lower solutions coloured yellow to indicate solutions in optima; this is a positive indication that the model is working as intended. There is almost a linear boundary (except for a few solutions close to the boundary), as we would expect from looking at the problem.

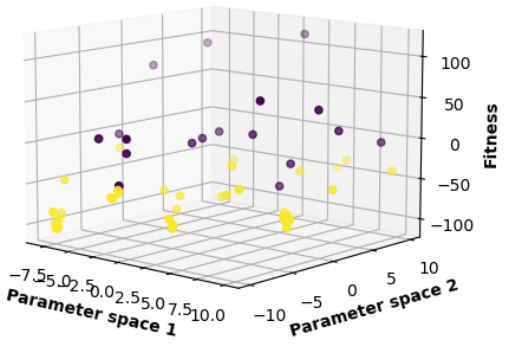


Figure 5.6: A single EA run on the Bird problem in the 2-D parameter single objective landscape. The EA evaluated for 100 generations generating an F_1 score of 0.88. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal.

We see that many of the solutions in optima have made much smaller perturbations (appear more clustered) than non-optima solutions and, therefore, may provide an indication that these optima prediction models are considering perturbation sizes to predict optima. We see how this model generally allows one to predict the solutions in optima and hence the locations of optima with a degree of accuracy. Visualising a single run may be useful for some applications, but we shall visualise multiple runs (the whole testing dataset) in the same figure for the remaining work. This creates a more intuitive visualisation and allows one to more easily consider and visually evaluate multiple runs in one figure.

5.2.2 Post-Encoder Visualisation

In Figure 5.7, solutions generated by the EA are plotted. The top row contains the raw output prediction data for the Styblinski testing problem. The model has been trained on the Bird problem data, and metrics can be found in the figure caption. We use the metrics as an approximation of model goodness, and we use the visualisation to determine the model's efficacy visually. We can see the four locations of optima indicated as yellow clusters; however, because of the non-exact methodology of finding optima during the data labelling stage and the fact that this is a statistical method, we expect some errors in classifying the data, which are likely carried forward into the model. Hence, not every single solution prediction is correct, but the distribution of solutions should be. As any error is prominent in the visualisation application, we produce an encoder to consider the distribution of solutions for each solution and recolour the solution if the solution prediction is not the same as the distribution of its neighbours. The encoder, therefore, has the effect of smoothing the prediction. That is, if a solution is predicted as non-optimal and the majority of its nearest neighbouring solutions are predicted to

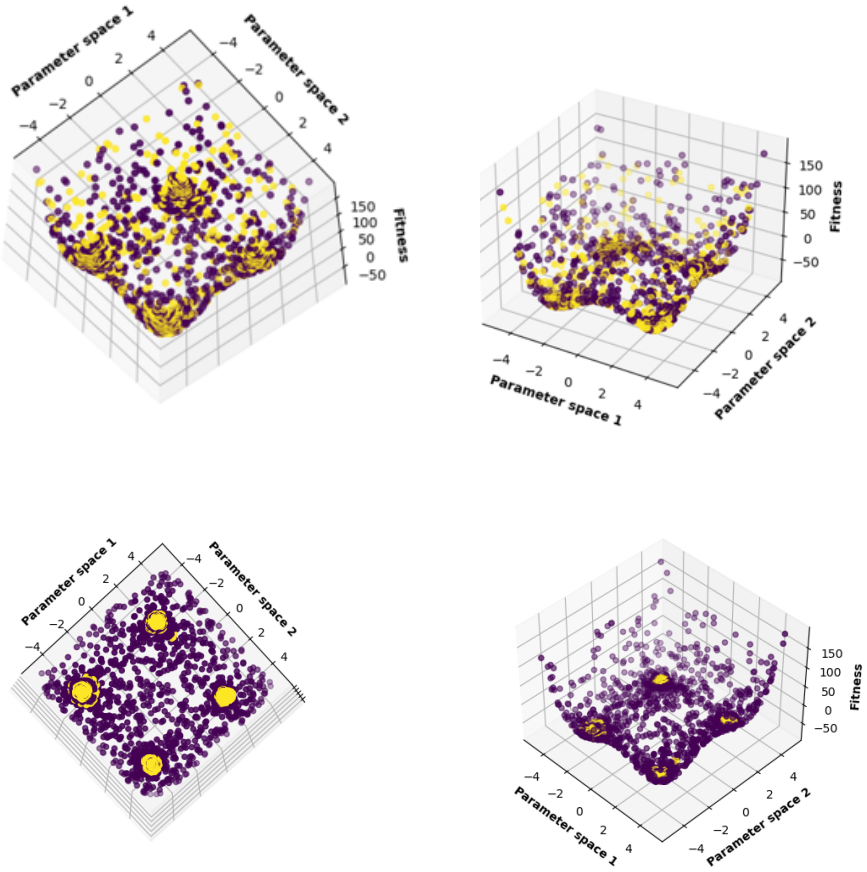


Figure 5.7: Trained on the Bird problem data and tested on the Styblinski problem data. The visualisation was created by setting $k = 50$. The testing data $F_1 = 0.92$. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal. The top row contains the raw output prediction data. The bottom row contains the prediction data after the smoothing encoder is applied.

be optimal, then the solution prediction will change to non-optimal status.

In the bottom row of Figure 5.7, we see the same data with the encoder applied to the predictions. For the encoder, there is a single parameter κ to set, which defines the number of neighbouring solutions to calculate the mean prediction (which therefore must be an integer value). This could also be considered the level of smoothing; if the values of κ increase, the greater the smoothing. Therefore the κ value is problem dependent, usually in the range of 1-150 for the problems used in this chapter. We set 10 as the default parameter unless otherwise stated; generally, as the ruggedness of the problem increases (a decrease in the auto-correlation and hence optima closer together), the smaller the values of κ. We apply this encoder to all visualisations from this point onward. The encoder does not affect the metrics. We see an F_1 score of 0.92 made up of a recall of 0.90, that is, 90% of the occurrences a solution was in optima it was predicted correctly, and a precision of 0.94, meaning of the predicted optimal solutions, 94% were correct and only 6% incorrect.

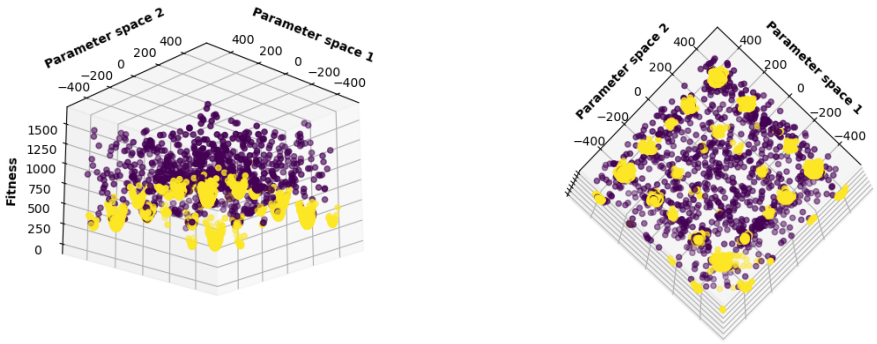


Figure 5.8: The model has been trained on the Bird problem dataset and tested on the Schwefel problem dataset, providing $F_1 = 0.85$. The visualisation was created by setting $k = 150$. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal.

In Figure 5.8, solutions generated by the EA are plotted and coloured based on the optima prediction model. The training problem was Bird, and the testing problem was Schwefel. Figure 5.9 considered a model trained on a search of the Ackley problem and used to make optima predictions on the Bird problem. The final figure, Figure 5.10, single objective problem model is a model trained on Ackley problem data and tested on the Hölder tray problem data.

As illustrated, the non-optima classed solutions are coloured purple, and the optima classed solutions are coloured yellow. We can see from all the figures that all the solutions that fall into optima are visually correct (that is, optima are identifiable from the visualisation colouring). The significant benefit of introducing this method in a single objective with two parameters is that we can see the optima as the troughs or minimums of the problem in the figure. We see, as expected, all the optima minima coloured correctly. Thus, the model's ability to predict optima for the problems are effective even though the testing and training problems appear to have little similarity in terms of problem features. The number of optima, locations and depth of optima varies for each problem. We see the testing problems generated strong F_1 scores: a score of 0.85 for Figure 5.8, 0.87 for Figure 5.9, and then a score of 0.88 for Figure 5.10. From a visual perspective, some solutions in very shallow minima are hard to detect visually.

Finally, we consider measuring all of the train/test problem combinations, and display the metrics in Table 5.2. For this experiment, we consider developing 30 models of the train/test combination (we choose a significantly large sample size of 30 so that the reader can perform any statistical tests they may wish); we then provide the mean F_1 score along with the standard deviation. When the training and testing problems are the same, the model has been tested on unseen (non-trained) data of the same problem; effectively, this is a validation set of the same size as the testing dataset. From the metrics, it appears some train/test combinations are better than others, perhaps owing to some similarity between how an EA perturbs solutions through the search space. For most problems, we see strong results. We expected larger differences between

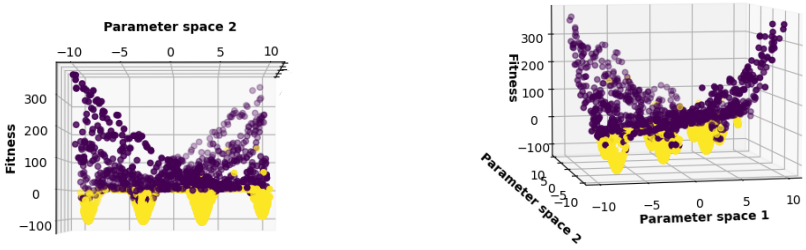


Figure 5.9: The model has been trained on the Ackley problem dataset and tested on the Bird problem dataset, providing $F_1 = 0.87$. The visualisation was created by setting $k = 50$. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal.

more dissimilar problems, which may indicate the EA performance between problems is more similar than expected, and the optima prediction performance may be highly dependent on the EA used. Using more complex EAs may require larger models to capture more complex patterns.

	Train Problem							
Test Problem	Ackley	CrossInTray	Himmelblau	Bird	Rastrigin	Styblinski	Schwefel	HolderTable
Ackley	0.73 ± 0.00	0.54 ± 0.02	0.93 ± 0.03	0.87 ± 0.01	0.83 ± 0.04	0.92 ± 0.03	0.82 ± 0.01	0.88 ± 0.01
CrossInTray	0.69 ± 0.13	0.54 ± 0.10	0.86 ± 0.17	0.83 ± 0.16	0.76 ± 0.15	0.85 ± 0.17	0.78 ± 0.15	0.82 ± 0.16
Himmelblau	0.73 ± 0.00	0.54 ± 0.01	0.94 ± 0.01	0.88 ± 0.00	0.83 ± 0.02	0.93 ± 0.01	0.82 ± 0.00	0.88 ± 0.00
Bird	0.73 ± 0.00	0.55 ± 0.02	0.92 ± 0.02	0.87 ± 0.01	0.81 ± 0.03	0.91 ± 0.02	0.82 ± 0.00	0.87 ± 0.01
Rastrigin	0.72 ± 0.00	0.50 ± 0.00	0.96 ± 0.00	0.87 ± 0.00	0.92 ± 0.01	0.94 ± 0.00	0.82 ± 0.00	0.89 ± 0.00
Styblinski	0.73 ± 0.00	0.54 ± 0.02	0.94 ± 0.02	0.88 ± 0.00	0.83 ± 0.03	0.92 ± 0.01	0.82 ± 0.00	0.88 ± 0.01
Schwefel	0.73 ± 0.00	0.55 ± 0.02	0.92 ± 0.02	0.87 ± 0.00	0.81 ± 0.03	0.91 ± 0.02	0.82 ± 0.00	0.88 ± 0.01
HolderTable	0.73 ± 0.00	0.55 ± 0.01	0.92 ± 0.02	0.87 ± 0.00	0.81 ± 0.02	0.91 ± 0.02	0.82 ± 0.00	0.88 ± 0.01

Table 5.2: The evaluated F_1 score means (μ) and standard deviations (σ) for different combinations of training and testing problems, evaluated over a sample size (number of models per problem) of 30.

For this work, we aim to demonstrate a concept: that predictive machine learning models can be used to enhance EAs. It is therefore important that we can generalise and simplify where possible to show this concept, which includes not optimising and utilising a large number of different hyper-parameters (for a large number of problems). Hence, we do not aim to create State-Of-The-Art results for every model but rather generalise and show that learning is possible and still effective with default hyper-parameters. The accuracy obtainable for each model is likely to be improved by hyper-parameter tuning, and so we would encourage the tuning of hyper-parameter models in practice (where time permits). However, to demonstrate with simplicity over a large number of different models, we keep the hyper-parameters constant.

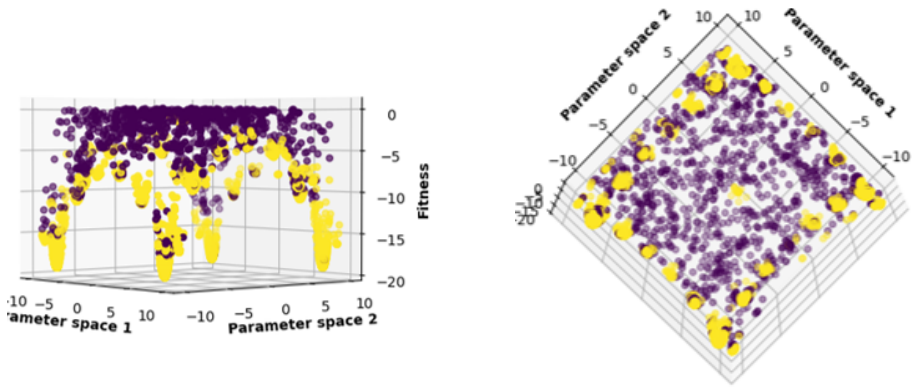


Figure 5.10: The model has been trained on the Ackley problem dataset and tested on the Hölder problem dataset, providing $F_1 = 0.88$. The visualisation was created by setting $k = 10$. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal.

5.3 Multi-Parameter Intelligent Optima Detection

By considering this application with dual-parameter single objective problems, we are able to evaluate the effectiveness of this methodology as we have direct visual data that can be plotted in three dimensions. In these instances, the minima are easily detectable by eye, which would not be the case in high-dimensional cases. However, for use in many real-world applications, it will be necessary to create models which can detect optima in multi- and many-dimensions, which is significantly more challenging to observe. This is because many problems scale greater than dual-parameter single-objective cases, and for the most beneficial application of this visualisation work, the visualisation should be able to detect solutions in optima in high dimensions by reducing to a manageable dimensional space as this would be highly insightful.

In this chapter, we aim to provide a framework to extend these models to multi- and many-parameter problems. The visual evaluation for these cases is more difficult as we can no longer identify optima regions as the minimum values in the z-axis. Instead, metrics are more greatly relied upon, like most ML work. The framework aims to change as little as possible between the dual-parameter and multi-parameter results to show a proof of concept that this work is possible rather than focusing on optimising the model's hyper-parameter to make incremental gains in accuracy. The multi-parameter work is more challenging as it requires the machine learning algorithm to receive a great amount of data to train on - this usually results in a greater training time and sometimes less accurate results from more noise and more complex patterns. One change we made was to extend the number of epochs from 30 to 60 to find more complex patterns.

We aim to train and evaluate these models and the methodology in three-, four- and five-parameter problems. We can evaluate the model accuracy using metrics such as the F_1 metric; however, evaluating the visualisation effectiveness is far more complex due to the difficulties in visualising high-dimensional solutions. In this work, we will vi-

sualise the data with the 'true' labels to colour the solutions and the data using the predictive model to colour the solutions. We use the term 'true' labels lightly - the reality is that the methodology captures key patterns but introduces noise into the labels. Therefore the label is not a 100% accurate truth but rather what the methodology considers truth (see the sampling method in Subsection 5.1.1). This means that the models are trained on data containing noise amongst the labels, so there will be compounded noise in the final predictive model. However, suppose metrics and the visualisation of the predictive models reflect the same patterns and accuracy as the training dataset. In that case, this will provide a strong indication that these models can be effectively used to predict solutions in regions of optima. At the top row of each figure, one will see the predictive output and the bottom row will display the 'true' output. We later discuss ways of optimising the data acquisition to reduce noise, effectively capturing more accurate training and test labels, and likely gaining better results, but for simplicity, the following methodology and results still achieve our objectives of demonstrating this proof of concept that an ML model can be trained to learn EA characteristics used to predict when solutions are optimal. For many of these problems, large amounts of the solutions are in optima - balancing solution distribution across the search space could show better results. For evaluation, we considered the extendable problems of the previous subsection, which are Ackley, Rastrigin, Schwefel and Styblinski, which can all scale arbitrarily in the parameter space.

5.3.1 Three-Parameter Results

In this subsection, we extend the dual-parameter single-objective problem solution data to three-parameter single-objective problem data and evaluate these results. We train the model in Figure 5.11 on the Rastrigin data and test on an unseen sample of the Rastrigin data to check whether this method will work for at least similar problem datasets. We see in Figure 5.11 an evident training process can be observed, the loss metrics (brown and dark blue line) are decreasing whilst the accuracy metrics are increasing. The model appears to learn most during the first ten epochs and then appears to make small gains.

We then plot the visualisation of the Styblinski data and colour according to the model results in the top row of the figure, and we plot the dataset and colour according to the labels in the bottom row - the juxtaposition of figures can be seen in Figure 5.12. We see in the bottom row (truth colouring) there is some noise, but generally, seven clusters of optima can be seen (yellow solutions) with non-optima solutions scattered between the solutions; from a visual perspective, the model appears to easily find the solutions in optima and the regions explored by the solutions not in optima. From the metrics, we gain an F_1 score of 0.90, indicating a high predictive accuracy for this model for this problem.

In Figure 5.13, we now evaluate a model trained on one problem (Schwefel) and tested on a dataset from a different problem (Rastrigin), this can be observed in Figure 5.13. Generally, the distribution of solutions in optima in the predictive visualisation appears correct when compared with the visualisation on the bottom row coloured as truth. We can see more non-optima labelled solutions in the more negative y_1, y_2 axes and more optima toward the denser regions of the solutions. This would be logical as an EA is naturally biased toward converging to optima, so more solutions are likely to lie in

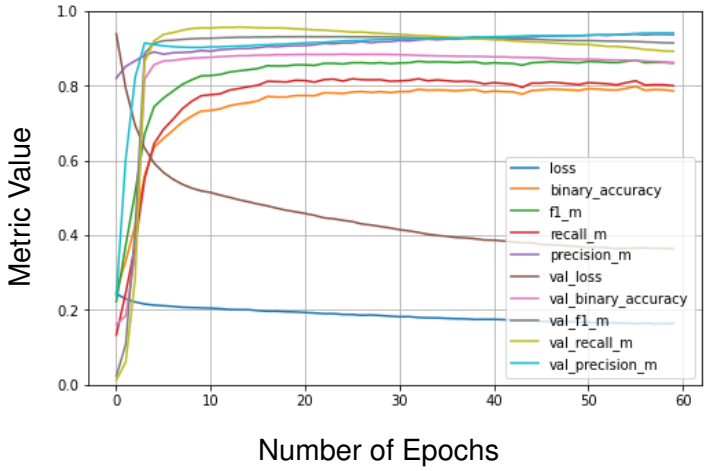


Figure 5.11: Training a model using Rastrigin data. The x-axis displays time, and the $y-$axis displays the metric value.

those stronger regions. Furthermore, the F_1 value of this model is 0.80 providing a strong indication that for the large majority of solutions, the label is correct. The major difference between the predictive model on the top row and the truth on the bottom row is that the predictive model tends to slightly overpredict non-optima; therefore, further training on the model could be conducted to get better results. However, using this visualisation, the DM should be able to determine which regions of the search space the optima lies and see the general distribution of solutions in regions of optima/not in regions optima. For individual solutions predictions the model has correctly predicted 85% of solutions correctly, $((TP+TN)/(FP+FN+TP+TN)) = ((1067+3189)/(376+368+1067+3189)) = 0.8512$.

In Figure 5.14 we see a much more effective model, both visually more similar to the truth colouring on the bottom row and a model possessing stronger metrics than in Figure 5.13. This could be because the problem types are more similar, or the training process was more effective for this instance. In the visualisation, it is more challenging to spot dissimilarities between the truth and prediction model colouring. Toward the lower y_1 upper y_2 axes, we see more solutions in non-optima states; toward the upper y_1, mid y_2 axes, we see a front of solutions indicating more optima. This model appears to be very effective at predicting solutions in regions of optima from both metrics and visual indicators perspectives, providing strong evidence that supports the hypothesis that ML models can be used to predict the characteristics of an EA, which in the case of visualisation allows one to observe optimal and non-optimal solutions in higher dimensions where this task is usually very difficult.

In Figure 5.15, we have trained on the Styblinski problem dataset and tested the model on the Schwefel problem dataset. In the top row, we see the predictive optima colouring. We see the predictive model over-predicting optima when compared to the truth colouring in the bottom row. We have chosen the visualisation runs randomly to get a good spread of performance; therefore, we have seen both better and worse results for this combination of training and testing. Compared to the previous two visualisations,

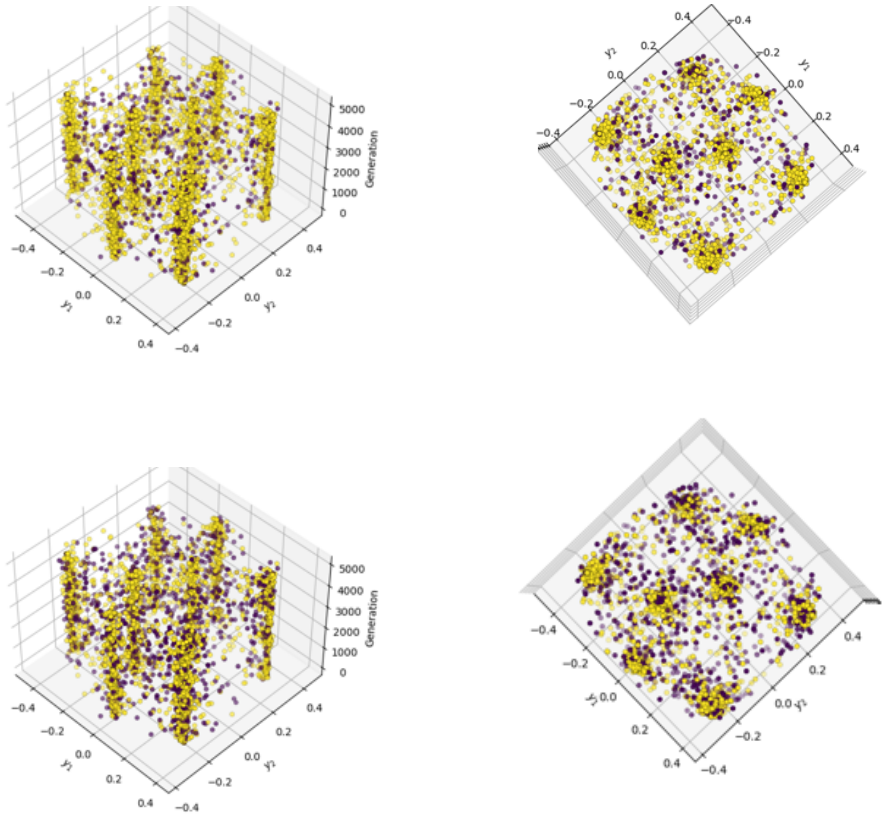


Figure 5.12: The model has been trained and tested on the Styblinski problem dataset, providing $F_1 = 0.90$. The visualisation was created by setting $k = 0$. The top row provides the predicted optima colouring, and the bottom row is the 'true' colouring. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal.

this result is more difficult to interpret and does not have high predictive accuracy, unlike the other two previous results. We note this is also reflected in the accuracy metric providing the model with an F_1 score of 0.78. We chose to include the poorer results to show that not all combinations of train/test problems provide a high degree of accuracy - however, most combinations produced a good or better than guess prediction of optima. Whilst re-training these poorer models with different model parameters can improve the results, it appears that there are combinations of problems that train faster and predict with higher accuracy - this is likely due to similarities between how the solution evolves through the landscape and the amount of noise introduced during the data acquisition/methodology stage of the process. Furthermore, a more complex problem would likely increase the amount of noise captured in the dataset. The qualitative results for the three-parameter problem combinations are displayed in Table 5.3 and support the qualitative analysis.

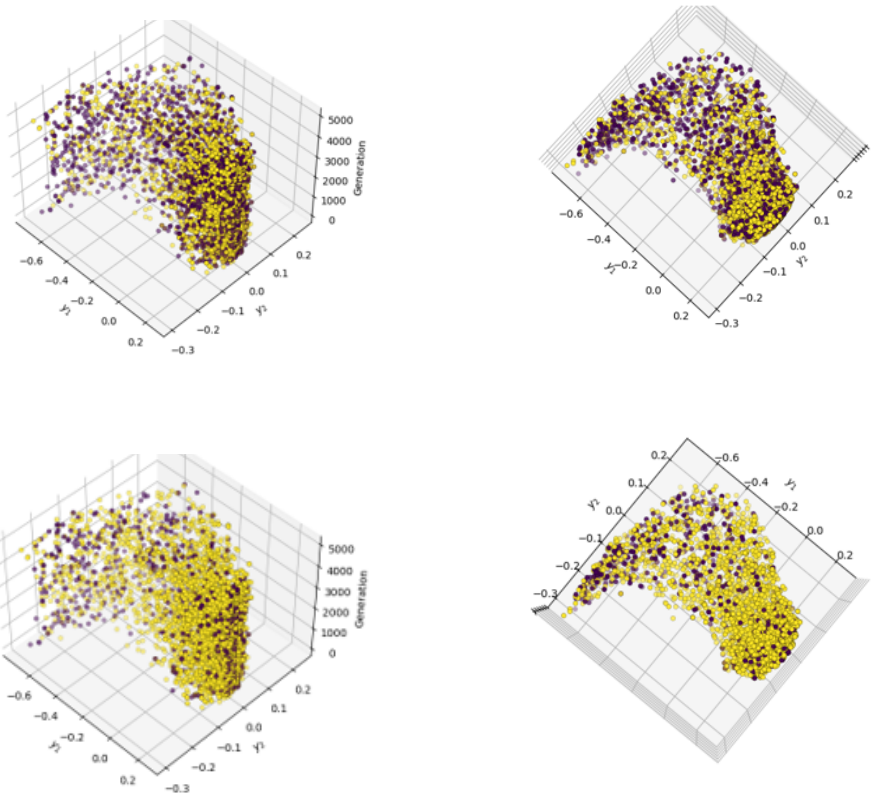


Figure 5.13: The model has been trained on the Schwefel problem dataset and tested on the Rastrigin problem dataset, providing $F_1 = 0.80$. The top row provides the predicted optima colouring, and the bottom row is the 'true' colouring. The visualisation was created by setting $k = 0$. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal.

	Train Problem			
Test Problem	Ackley	Rastrigin	Styblinski	Schwefel
Ackley	0.80 ± 0.11	0.83 ± 0.14	0.84 ± 0.15	0.73 ± 0.12
Rastrigin	0.80 ± 0.01	0.88 ± 0.01	0.91 ± 0.00	0.78 ± 0.00
Styblinski	0.68 ± 0.13	0.73 ± 0.14	0.85 ± 0.16	0.74 ± 0.14
Schwefel	0.69 ± 0.04	0.74 ± 0.04	0.87 ± 0.03	0.76 ± 0.01

Table 5.3: The evaluated F_1 score means (μ) and standard deviations (σ) for different combinations of training and testing problems with three parameters, evaluated over a sample size (number of models per problem) of 30.

5.3.2 Results for Larger Numbers of Parameters

In this subsection, we look at extending the visualisations for use with four- and five-parameter problems and evaluate the performance of the models and the effectiveness of the visualisation. As we extend to additional parameters, we may expect to see more complexity which could require the model to train for longer to achieve similar accuracy

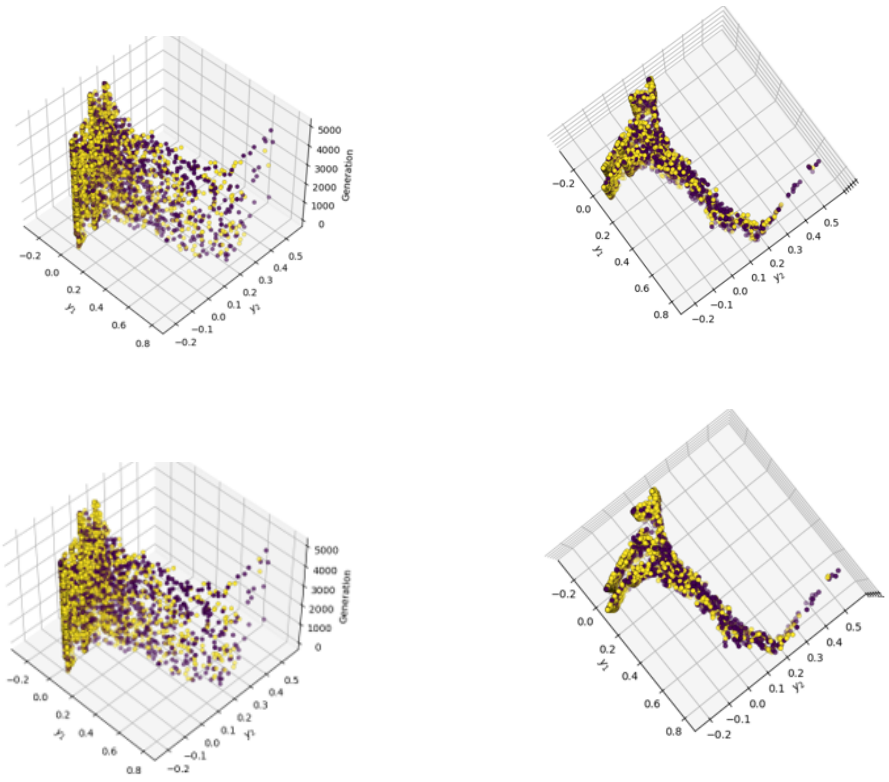


Figure 5.14: The model has been trained on the Schwefel problem dataset and tested on the Styblinski problem dataset, providing $F_1 = 0.90$. The top row provides the predicted optima colouring, and the bottom row is the 'true' colouring. The visualisation was created by setting $k = 0$. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal.

as a lower parameter model; alternatively, adding more data may also improve the model accuracy by providing more possibilities of pattern detection (the sum of the perturbations are larger). We note as the dimensionality of the solutions is further reduced, the accuracy of the data with respect to intergenerational distances also decreases. This can be seen by the four-parameter results in Table 5.4. As one would expect from an observation of the table, the visualisation produced are similar to the three-parameter results, and therefore we omit these visualisations to reduce repetition. However, we include both the quantitative and qualitative analysis of the two-, three- and five-parameter results to ensure the reader has the evaluation over a good spread of results.

We extend this work to the creation and evaluation of five-objective data-trained models. We can see in Figure 5.16 the visualisations from a model trained and tested on Rastrigin. We see the truth colouring visualisation on the bottom row fairly accurately represented in the predictive model colouring on the top row. We see an increasing amount of optima as the y_2 value increases in both spaces, and so the general distribution of optima is well reflected by the predictive model. The F_1 score has a high

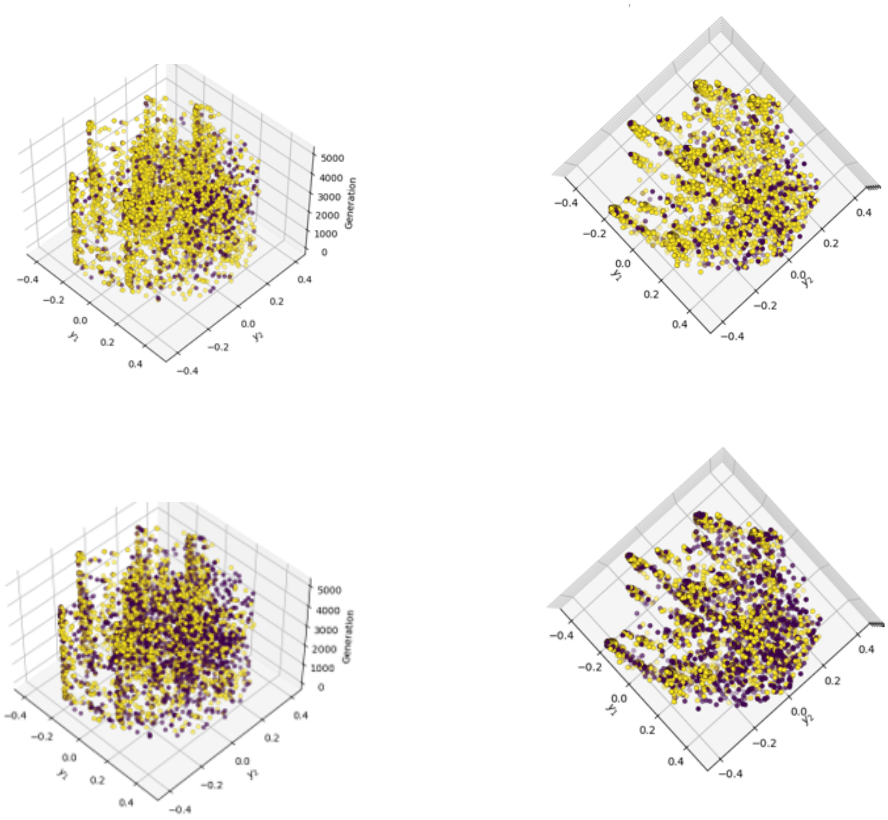


Figure 5.15: The model has been trained on the Styblinski problem dataset and tested on the Schwefel problem dataset, providing $F_1 = 0.78$. The top row provides the predicted optima colouring, and the bottom row is the 'true' colouring. The visualisation was created by setting $k = 0$. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal.

accuracy of 0.9, which indicates strong predictive power for this model. As in previous visualisations, we see the less dense regions yielding more non-optima (purple solutions), and this appears intuitively correct as the EA's objective is to converge on optima; the data will therefore be biased at containing data points in or around optima.

	Train Problem			
Test Problem	Ackley	Rastrigin	Styblinski	Schwefel
Ackley	0.73 ± 0.02	0.78 ± 0.04	0.87 ± 0.01	0.71 ± 0.00
Rastrigin	0.75 ± 0.01	0.84 ± 0.02	0.88 ± 0.00	0.72 ± 0.00
Styblinski	0.69 ± 0.02	0.71 ± 0.03	0.86 ± 0.01	0.70 ± 0.00
Schwefel	0.70 ± 0.03	0.73 ± 0.05	0.85 ± 0.03	0.70 ± 0.01

Table 5.4: The evaluated F_1 score means (μ) and standard deviations (σ) for different combinations of training and testing problems with four parameters, evaluated over a sample size (number of models per problem) of 30.

In Figure 5.17, we train on the Styblinski problem dataset and test on the Rastrigin

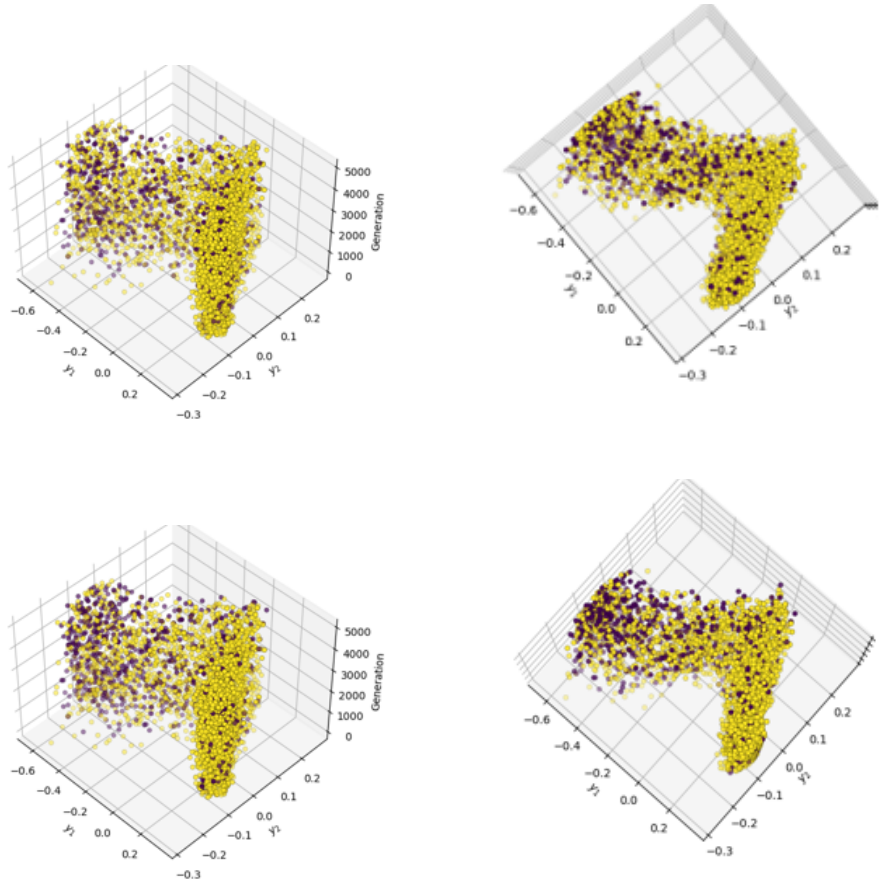


Figure 5.16: The model has been trained and tested on the Rastrigin problem dataset, providing $F_1 = 0.90$. The visualisation was created by setting $k = 0$. The top row provides the predicted optima colouring, and the bottom row is the 'true' colouring. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal.

dataset. We see the predictive model coloured visualisation has captured the denser areas of non-optima (purple solutions) toward the northern region of the figure well. However, the model does seem to underpredict optima slightly, with more purple solutions than expected.

Figure 5.18 shows a model trained on the Styblinski problem dataset and tested on the Schwefel problem dataset. Unfortunately, the result of this model is not so accurate. This is an example of the model being ineffective. There are several possible reasons for this, such as too much noise in the dataset, the dissimilarity between problems, or ineffective model training (e.g., the model may need to be trained for longer). We can see from the visualisation that the model over-predicts solutions to be in optima. The F_1 metric value of 0.54 is also reflecting this poor result. However, one can see the distribution of optima from the more yellow area toward the lower y_1 values, and so whilst difficult, there does appear to be a small amount of learning. Whilst most of the models considered so far appear to work well, this example shows that it is not

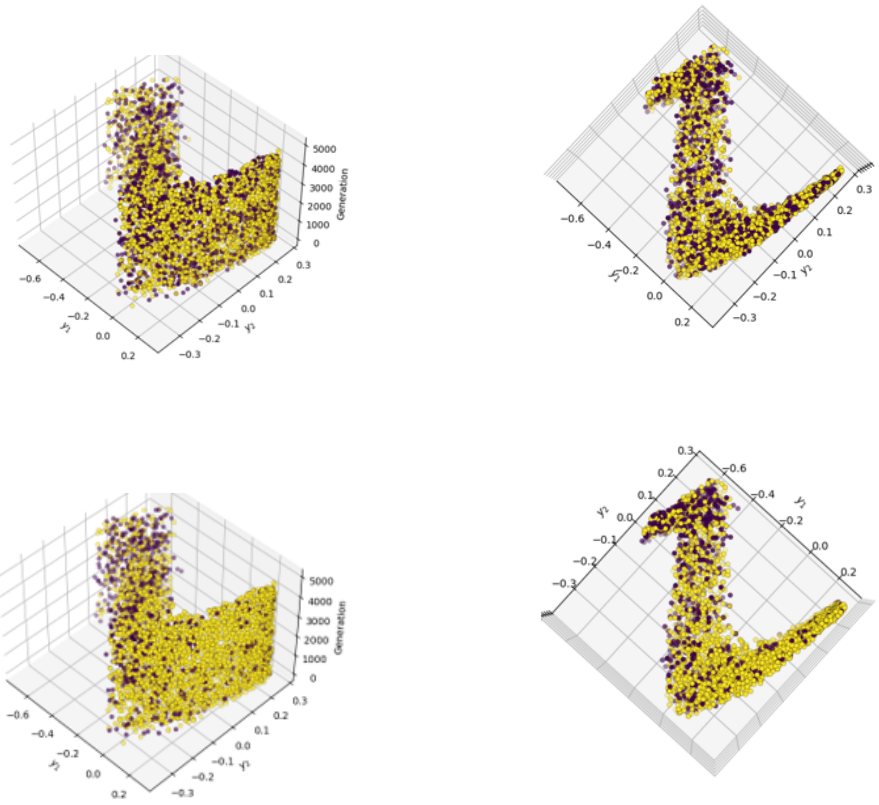


Figure 5.17: The model has been trained on the Styblinski problem dataset and tested on the Rastrigin problem dataset, providing $F_1 = 0.79$. The top row provides the predicted optima colouring, and the bottom row is the 'true' colouring. The visualisation was created by setting $k = 0$. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal.

the case that all training testing combinations will produce accurate results. We later discuss possible causes of this phenomenon.

Finally, Figure 5.19 shows a visualisation coloured using a model trained on the Schwefel problem dataset and tested on the Rastrigin problem data. We see an overprediction of non-optima; however, the general distribution of non-optima can be seen from the visualisation; the lower part of the space clearly contains fewer optima, as seen as a lower distribution of purple colouring. Looking at the visualisation, one can see which regions of the space are more optimal (the north). Whilst a better result than the previous visualisation, we have seen more accurate train/test combinations in this work. The F_1 metric returns a value of 0.86.

In Table 5.5, we can observe the five parameter results. We can see the different train and test problem combinations. Comparing with the three- and four-parameter results, we can see as the number of parameter values increases, the F_1 accuracy decreases and this is reflected in the qualitative analysis provided for the visualisations.

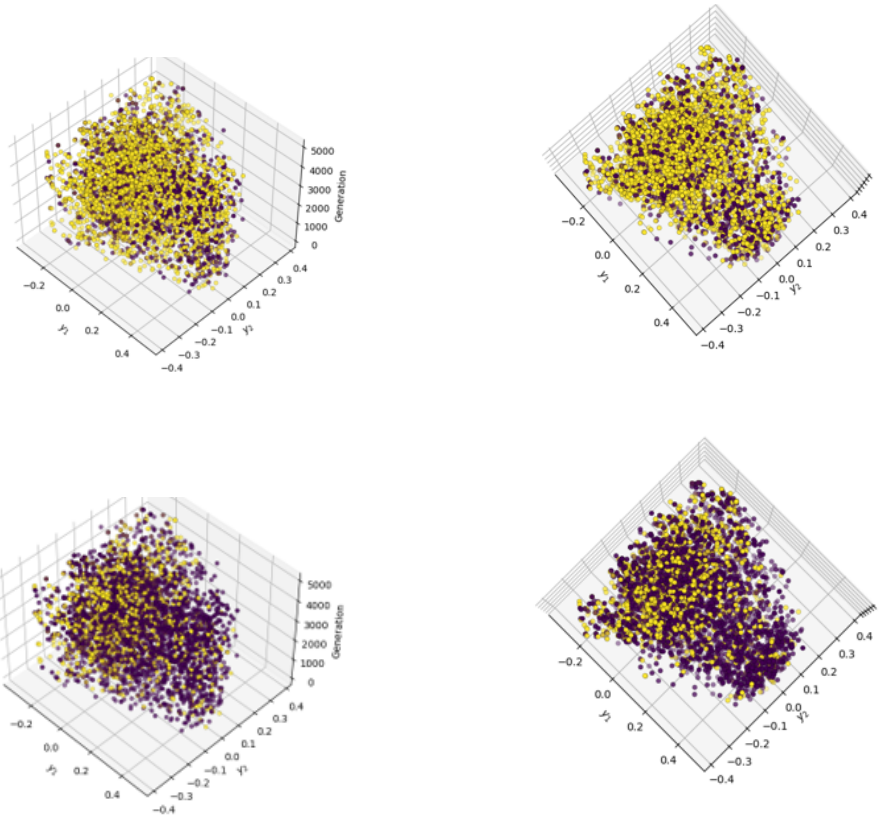


Figure 5.18: The model has been trained on the Styblinski problem dataset and tested on the Schwefel problem dataset, providing $F_1 = 0.54$. The top row provides the predicted optima colouring, and the bottom row is the 'true' colouring. The visualisation was created by setting $k = 0$. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal.

	Train Problem			
Test Problem	Ackley	Rastrigin	Styblinski	Schwefel
Ackley	0.62 ± 0.01	0.76 ± 0.03	0.29 ± 0.00	0.57 ± 0.00
Rastrigin	0.64 ± 0.01	0.83 ± 0.02	0.29 ± 0.00	0.57 ± 0.00
Styblinski	0.62 ± 0.02	0.79 ± 0.06	0.29 ± 0.00	0.57 ± 0.01
Schwefel	0.58 ± 0.04	0.68 ± 0.06	0.28 ± 0.01	0.57 ± 0.01

Table 5.5: The evaluated F_1 score means (μ) and standard deviations (σ) for different combinations of training and testing problems with five parameters, evaluated over a sample size (number of models per problem) of 30.

5.3.3 Model Insight

We now consider obtaining insight into the processes of these models. In this work, we use an explainable AI method denoted SHAP (more details can be found in Section 2.3). After computing the SHAP values, indicating the influence each feature has, we plot the values. We compute the SHAP values for the three- and five-parameter

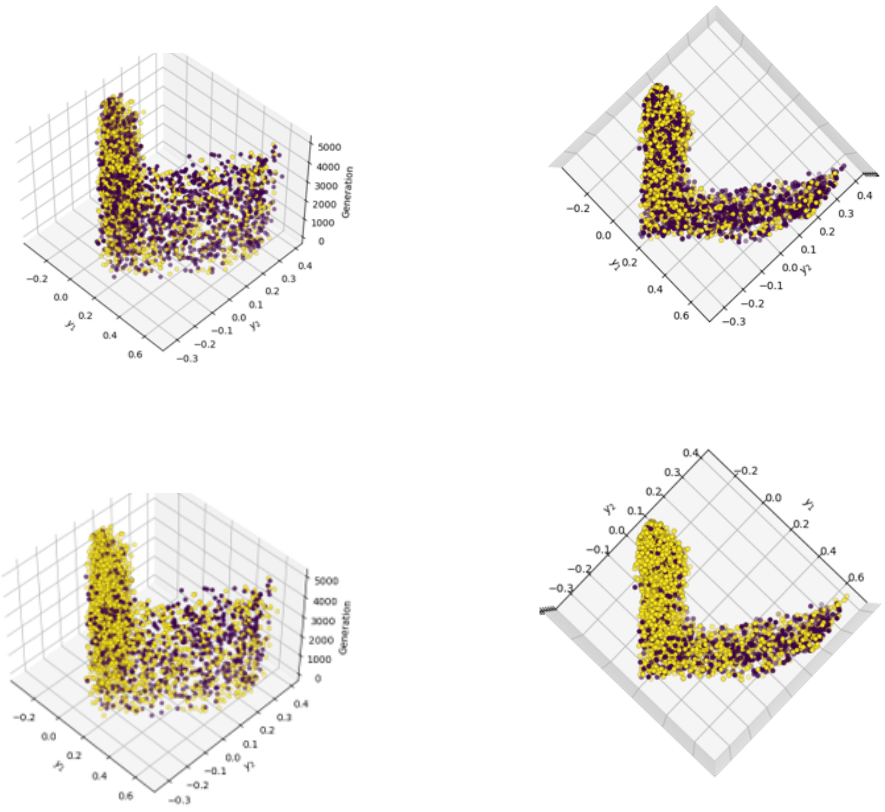


Figure 5.19: The model has been trained on the Schwefel problem dataset and tested on the Rastrigin problem dataset, providing $F_1 = 0.86$. The top row provides the predicted optima colouring, and the bottom row is the 'true' colouring. The visualisation was created by setting $k = 0$. The ● solutions are predicted non-optimal, and the ● solutions are predicted optimal.

problems. The SHAP values are computed on time series EA runs with the same data structure as the model input, e.g., arrays of size 10×2.

In this analysis of Figure 5.20 we see the SHAP values for a model trained on the Ackley problem dataset in the bottom row; we can determine that the parameter space has nearly twice the impact effect on the model than perturbations in the objective space. We can also see the objective space having increasing importance on the model prediction as the number of parameters increases and the problem becomes more complex. Interestingly, in the bottom row, we see SHAP values for the Styblinski problem dataset, and one can observe an opposite effect where the objective space appears to have greater attribution to the model. However, as the complexity of the problem increases, as we see in the 5-D Styblinski model, the objective space has an increasing effect on the model. Therefore, we conclude that the attribution of each feature depends on the problem model. The feature attribution may also be affected by the curse of dimensionality. For some problems, the parameter space perturbations

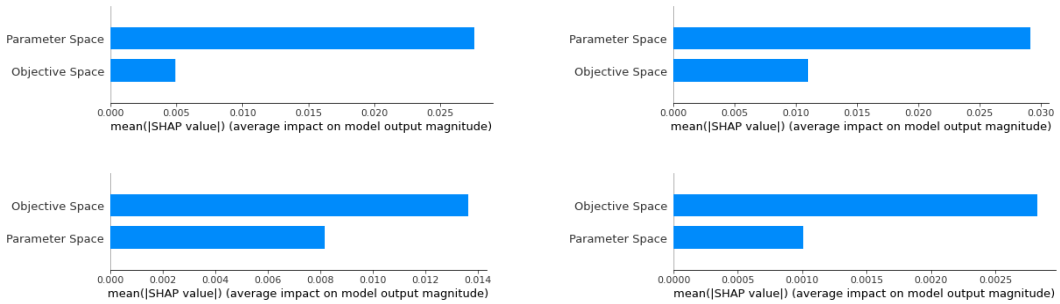


Figure 5.20: The SHAP values for a model trained on the Ackley problem dataset are in the top row. In the bottom row, the SHAP values for a model trained on the Styblinski problem dataset. The left column contains 3-D model SHAP values, and the right column contains the 5-D model SHAP values.

appear to hold a greater effect on the model predictions and vice versa.

For further work, one may consider other features to input into the model, which could provide additional information for the decision process. Therefore, we recommend hypothesising informative features to input into model predictions and then testing the hypothesis by using SHAP to determine the feature's importance and accuracy.

5.4 Conclusion and Future Work

In this chapter, we aim to develop a novel way of capturing EA characteristics to create a predictive model of EA performance using ML. There are many potential applications for this development, including using the model to engineer a self-adaptive parameter EA, where the parameters (such as mutation and crossover) can be adapted with regard to algorithm performance (e.g., whether the solutions are challenged by regions of local optima), In this chapter, we demonstrate the ability to create intelligent visualisations. Intelligent visualisations are visualisation created using AI. In this case, we use the predictive models to colour the solutions based on whether the model predicts a solution to be in optima.

In this chapter, we have a detailed methodology to acquire relevant training data, build the ML model and train on the data; this methodology was sufficient for predicting optimal solutions correctly for the problems and dimensions we tested. We have also seen these predictive models trained and evaluated on many different combinations of problem data. However, the models are not all likely to be in an optimal state (due to a lack of hyper-parameter search), and combinations of problems used in this work were chosen arbitrarily to ensure we evaluate a full unbiased spread of results, although; generally, the predictive models and hence the visualisations have been effective at showing which solutions are in optima even in higher dimensional space due to the implementation of an LMDS visualisation from the previous chapter complimented by colouring from these predictive models.

The intelligent visualisation, for the problems in the results of this work, may allow one to observe which solutions are in optima and not in optima. The models also may allow one to observe the general distribution of the optima and see which regions

of the search output optima and non-optima to gain a better insight into the search. However, this is only confirmed by the author's analysis and could later be evaluated with a usability study. From signals in the visualisation, such as EA solution bias, and providing a visualisation of the true space to compare with the predictive visualisation, we can confirm the models are relatively accurate visually. This is also backed by the strong balanced metric F_1 score yielded by most of the models, and the models with the lower F_1 score often yielded poor visual results.

The models may be able to accurately predict optima enough to be used as a visualisation tool; however, one needs to collect training data to create the models. We note that the ML model is a statistical approach rather than an exact approach, creating a trade-off in accuracy and computation time. Therefore one may wonder why use the ML colouring models if we can use the sampling methodology to colour directly. However, once the model is trained, the sampling method is much slower than forward propagation for these simple models; also, as the dimensional space increases, the number of samples required to check the space exponentially increases. It is also possible to not have access to the objective function, or the objective space may be dynamic.

Due to the novelty of this work, unfortunately, it opens up as many questions as answers, paving the way for advancements in this technology. Whilst we aim to communicate a proof of concept - that deep learning models can be used to predict EA characteristics - we keep the methodology as simple as possible whilst gaining satisfactory results. We also keep many model hyper-parameters constant. For achieving an optimal model for real-world use, one would likely take additional steps such as hyper-parameter tuning for greater predictive accuracy and could adapt the methodology more niche to their problem. From looking at the “truth" coloured visualisation, we note that the method we use likely introduces a level of noise into the dataset; therefore, when creating a predictive model, some of this noise may be introduced into the model. Consequently, in future works, the first major advancement would be to consider other methods of data acquisition (the capturing of data labels) to reduce the noise as much as possible. This may come from new methods or promising/well-tested existing methods in the optima detection literature. This reduction of noise in the data labelling would likely improve the accuracy of the models. We note, however, for simplicity in communicating this work, this methodology was sufficient, as seen by the results.

In future work, we would also like to see an extension to a greater number of objectives to increase the scalability of these models. Although we tested on six well-known benchmark problems, we would also welcome an extension to different, real-world, and more complex problems. For simplicity, we inputted into the machine learning algorithm a dataset which used a single column for the objective space perturbations and a single column for the parameter space perturbations (e.g., a dataset with two columns in total); it may be for more complex problems separating the columns into each parameter and each objective perturbation (also adding other non-training data-related features which could be used to learn EA characteristics) could improve the results for more complex/greater dimensional problems. But it may also add complexity with more training features which could take much longer to train at the reward of greater accuracy. We also note we kept the epochs for the multi-parameter visualisations fixed

at 60 to show how these models can still perform well with small training times/lack of training hardware, however for some of these problems, it is likely that a greater training time will yield better results providing one is careful not to overtrain the model.

Future work could also consider applying to different EAs; in this work, we limit the training to use a relatively simple (1+1)–ES; however, this would require an adaptation to the methodology for the data acquisition stage. A more complex EA may yield better results with greater training times and more complex ML models. A limitation of this work demonstrated by the results (such as Figure 5.18) is that some training/testing combinations do not work well, and others do work well. Some investigation into why some combinations are better than others is appropriate. We suspect much of this is down to a level of similarity between how solutions act during the search of a problem. One should also consider the possibility of investigating whether three-dimensional data-trained models work on five-dimensional data-trained models and the limitations of this extension. Finally, one could also consider the possibility of multi-problem training and testing - scaling this to some universal model which is trained on many diverse problems and can be applied to nearly any new test problem - perhaps with some limitations.

This chapter provides a proof of concept demonstrating how an ML model can be trained on a problem to learn EA characteristics (solutions in optima) and make predictions on a different problem providing many potential applications to enhance and support EAs. In the chapter, we apply it to develop an intelligent visualisation, using an ML model to make a prediction on whether the solution lies in optima. We discuss the potential limitations of this method. We demonstrate strong results both visually and with metrics. Due to this work's novelty, we can highlight a significant amount of future work, which could be an exciting future avenue for research.

Chapter 6

Conclusion

This thesis aimed to propose informative and explainable visualisation methods to assist with understanding the optimisation process. In this conclusion, we summarise the chapters and the main contributions of each chapter. We then discuss future work.

6.1 Visualising the Evolutionary Algorithm Search Process

Evolutionary algorithms can be difficult to comprehend and communicate, and visualisation is a proven tool as a medium for displaying information. Unfortunately, most of the literature only considers a small subset of the population (the approximation set), revealing information about the approximation front/set but precluding information about the search process. One way to capture information about the entire search process is to visualise the entire population, which is sometimes high-dimensional and much larger than the approximation set, making it non-trivial to visualise in an interpretable manner. The large amount of data to display can quickly become overwhelming, and solutions between generations can overwrite each other (a many-to-one mapping) and cluster together in a way which is difficult to interpret by the user. Dimension reduction is one attempt at alleviating the high-dimensional issues associated with high-dimensional solutions; however, the complexity of the most accurate dimension reduction techniques makes it challenging to reduce for large approximation sets, let alone whole populations. This limits these methods to simple problems that can be solved with a small number of function evaluations.

In Chapter 3, we created a new method to visualise the entire population in a reduced complexity way which could even be parallelised (computing each generation on a different core), allowing one to visualise the population of multi- and many-objective problems. Although this method scales well and shows a number of problem and algorithm artefacts, we note that improvements could be made to maintain a normalised orientation of the principal components. This would improve the interpretability of the visualisation. We, therefore, developed a second method utilising an MDS approximator to allow one to scale the visualisation to large populations and generations. We evaluated this work in depth. For the problems tested, the proposed method is visually akin to the previous parameter space method, requiring less than 1% of the time and memory necessary to visualise the same objective space solutions. This method allows one to create a scalable, high-accuracy visualisation of a multi- and many-objective population as it evolves through the search. The benefits to an EA user may range from better communication of algorithms and gauging the quality of the search to better interpretability, leading one to consider using this visualisation as a base for other applications within the EA domain.

We believe the primary future direction for this work is the application of this type of vi-

sualisation to enhance other areas of EC. For example, a significant amount of thought is going into explainability within the EA field (the ECXAI workshop being one such example), and some existing visualisations may be adapted to create explainability methods. Due to the wealth of information obtained about the process, these visualisations could also be used to communicate information about the search. With the metaphor issues obscuring similar/often identical 'novel' EAs present within the EA community (Sörensen 2015, Camacho-Villalón et al. 2022*a*,*b*, Aranha et al. 2022), one could visually compare the search of suspect EAs to identify the degree of similarity. Finally, with the scalability issues addressed, the improvements to an optimal visualisation would need to focus on accuracy. Therefore, one could consider the choice of landmarks and choosing specific landmarks that maximise the accuracy of the model. Furthermore, other methods in the Nyström algorithms class or fast linear affine transformations could be evaluated to see if the accuracy of the visualisation can be increased at the reduced complexity.

6.2 Visualisations for Explainable Evolutionary Computing

Most recently, questions about the explainability of evolutionary computing have risen to the forefront of research. We have seen how XAI has expedited the integration of AI into industry and vast academic domains. Requirements have risen regarding the need for explainability in EC, methods which could support the user's comprehension of the search via greater explainability of the algorithm-problem interaction, the effect hyperparameters have on the search, the robustness of solutions, the traceability of solutions and debugging potential. With greater explainability of the search comes greater trust in the methods and enhancements/developments of the methods from better understanding. In the year 2022, the term Evolutionary Computing for explainable AI (ECXAI) was coined at The Genetic and Evolutionary Computation Conference, covering explainability for EC and EC methods that could provide explainability to AI.

In Chapter 4, due to the novelty of explainability in EC, we motivate this new subdomain of EC. We then, as one of the earliest EC explainability methods, create a visualisation with the aim of creating explainability. The visualisations explainability aims to address the requirements outlined in the previous paragraph. We use the methodology in Chapter 3 as the basis to visualise high-dimensional solutions in a visually manageable 3-D space. We then adapt the methodology to show the effect of the reproductive operators and the traceability of solutions. We denote this EC explainability method, the Population Dynamics Plot (PopDP). We evaluate PopDP on a real-world offshore windfarm benchmark problem and analyse the explainability provided. We show the results of PopDP appear to be highly explainable and informative about the search process by illustrating the population of solutions, the parent-offspring lineage, the solution perturbation operators, and the path of the search process.

In future work, we suggest a community effort to define explainability within the domain of EC. This could be by proposing a set of criteria/questions about what an explainable method should reveal about the process (e.g., the regions of search space visited by the population); once a set of criteria has been created, one could weight the criteria according to importance, methods could then be quantitatively (number of criteria met) and qualitatively reviewed. These criteria should be defined by a diverse range

of users, such as researchers, EA developers, EA users and key stakeholders. From this work, we should consider the type and quantity of information to reveal about the EA process. Further work could also suggest applying PopDP to a diverse range of problems, including continuous, discrete, benchmark and real-world problems, with a heavy emphasis on the latter type. As more explainability methods come into existence, a review should be done into the benefits and comparisons of each. Finally in future work, a usability study should be undertaken with explainability methods like PopDP to support the conclusions of the work and further development into optimising the explainability of the methods. By creating better explainability methods, one may be able to better communicate, develop trust and enhance EC, with benefits to all stakeholders.

6.3 Deep Learning for Intelligent Visualisation in Evolutionary Computing

One of the critical challenges for an EA evolving solutions through a search landscape is the presence of global and local optima. Local optima, in particular, can cause premature convergence and suboptimal results. Detecting optima within a visualisation would be very informative and help assist with the detection of other features as well as justify the algorithm's performance. However, detecting local optima in visualisations of high-dimensional solutions is challenging because detection relies on secondary indications such as clustering of solutions and these clusters are not always accurately preserved when dimensions are reduced.

In Chapter 5, we investigate the use of machine learning to identify when solutions are in optima based on the characteristics and behaviour of the solution over time. The benefit of using machine learning is that once trained, the computation is fast, autonomous and can work without access to the objective function. There are many possible applications and demands for this work. We use the models to colour solutions in a visualisation based on whether the solution is predicted to be in optima. Particularly after the reduction in dimensions, defining which solution are in optima is a non-trivial task, as often the dimension reduction induces a reduction in the accuracy of the visualisation. We provide a suggested methodology and define the limitations of creating these models. We also test and evaluate these models on multi- and many-parameter problems. The work reveals an intelligent visualisation that reveals further information about the search, such as the regions of difficulty for the solutions, distributions and locations of optima. The visualisation also appeared an effective way to demonstrate the accuracy of these models, further supporting the use of visualisations for communicating algorithms in a slightly different context.

We recognise the limitations of this work in Chapter 5 and that this work is more of a proof of concept demonstrating that the key concepts work effectively. Therefore it is important that we outline a future projection of work for potentially overcoming these limitations. Future work is detailed in the Chapter 5 conclusion, but we summarise the key future work conclusions here. We first note that further work could be conducted to increase the scalability of these models to different problems, to greater numbers of objectives and parameters and different EAs. The models used in this work were kept relatively simple to communicate the proof of concept that ML models can be used to enhance EAs. However, when used for real-world problems, one may want to

optimise the models (such as with hyperparameter tuning) and adapt the methodology to reduce noise in the datasets (particularly during data labelling). One could also optimise the training time required. Further work should include an investigation of which train/test problem combinations are most effective and why. Work could also consider different data representations inputted into the model to improve accuracy or create better generalisability. A universal model which is trained on many diverse problem types and applicable to many different problem types would also be a valuable addition to the literature. Finally, other characterisations, predictions, and ways ML can be used to enhance EAs should be considered in future works.

6.4 Summary

This thesis addresses several research challenges by proposing visualisation methods to enhance interpretability and explainability within evolutionary computing. We have summarised the chapters and promising future work in this conclusion.

List of references

Adair, J., Ochoa, G. & Malan, K. M. (2019), Local optima networks for continuous fitness landscapes, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', pp. 1407–1414.

Alba, E. & Chicano, J. F. (2006), Evolutionary algorithms in telecommunications, *in* 'MELECON 2006-2006 IEEE Mediterranean Electrotechnical Conference', IEEE, pp. 795–798.

Andersen, H., Lensen, A. & Browne, W. N. (2022), 'Improving the search of learning classifier systems through interpretable feature clustering'.

Ang, K. H., Chong, G. & Li, Y. (2002), Visualization technique for analyzing non-dominated set comparison, *in* 'Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)', Vol. 1, p. 36.

Apley, D. W. & Zhu, J. (2020), 'Visualizing the effects of predictor variables in black box supervised learning models', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **82**(4), 1059–1086.

Aranha, C., Camacho Villalón, C. L., Campelo, F., Dorigo, M., Ruiz, R., Sevaux, M., Sörensen, K. & Stützle, T. (2022), 'Metaphor-based metaheuristics, a call for action: the elephant in the room', *Swarm Intelligence* **16**(1), 1–6.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R. et al. (2020), 'Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible ai', *Information fusion* **58**, 82–115.

Ashby, M. (2000), 'Multi-objective optimization in material design and selection', *Acta materialia* **48**(1), 359–369.

Asonitis, T., Allmendinger, R., Benatan, M. & Climent, R. (2022), Sonopt: Sonifying bi-objective population-based optimization algorithms, *in* 'International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)', Springer, pp. 3–18.

Audet, C., Bigeon, J., Cartier, D., Le Digabel, S. & Salomon, L. (2020), 'Performance indicators in multiobjective optimization', *European journal of operational research* .

Azami, H., Malekzadeh, M., Sanei, S. & Khosravi, A. (2012), 'Optimization of orthogonal polyphase coding waveform for mimo radar based on evolutionary algorithms', *Journal of mathematics and Computer Science* **6**(2), 146–153.

Bacardit, J., Brownlee, A. E., Cagnoni, S., Iacca, G., McCall, J. & Walker, D. (2022), The intersection of evolutionary computation and explainable AI, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', pp. 1757–1762.

Bäck, T., Preuss, M., Deutz, A., Wang, H., Doerr, C., Emmerich, M. & Trautmann, H. (2020), *Parallel Problem Solving from Nature-PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part II*, Vol. 12270, Springer Nature.

Bader, J. & Zitzler, E. (2011), 'Hype: An algorithm for fast hypervolume-based many-objective optimization', *Evolutionary computation* **19**(1), 45–76.

Barnes, J. & Hut, P. (1986), 'A hierarchical o (n log n) force-calculation algorithm', *nature* **324**(6096), 446–449.

Bernadó-Mansilla, E. & Garrell-Guiu, J. M. (2003), 'Accuracy-based learning classifier systems: models, analysis and applications to classification tasks', *Evolutionary computation* **11**(3), 209–238.

Beyer, H.-G. & Deb, K. (2001), 'On self-adaptive features in real-parameter evolutionary algorithms', *IEEE Transactions on evolutionary computation* **5**(3), 250–270.

Bezerianos, A., Dragicevic, P., Fekete, J.-D., Bae, J. & Watson, B. (2010), 'Geneaquilts: A system for exploring large genealogies', *IEEE Transactions on Visualization and Computer Graphics* **16**(6), 1073–1081.

Biederman, I. (1987), 'Recognition-by-components: a theory of human image understanding.', *Psychological review* **94**(2), 115.

Bishop, C. M., Svensén, M. & Williams, C. K. (1998), 'Gtm: The generative topographic mapping', *Neural computation* **10**(1), 215–234.

Bosman, P. A. & Thierens, D. (2003), 'The balance between proximity and diversity in multiobjective evolutionary algorithms', *IEEE transactions on evolutionary computation* **7**(2), 174–188.

Breiman, L. (2001), 'Statistical modeling: The two cultures (with comments and a rejoinder by the author)', *Statistical science* **16**(3), 199–231.

Brockhoff, D., Friedrich, T. & Neumann, F. (2008), Analyzing hypervolume indicator based algorithms, *in* 'International Conference on Parallel Problem Solving from Nature', Springer, pp. 651–660.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. et al. (2020), 'Language models are few-shot learners', *Advances in neural information processing systems* **33**, 1877–1901.

Brownlee, A. E., Epitropakis, M. G., Mulder, J., Paelinck, M. & Burke, E. K. (2022), 'A systematic approach to parameter optimization and its application to flight schedule simulation software', *Journal of Heuristics* **28**(4), 509–538.

Burlacu, B., Affenzeller, M., Kommenda, M., Winkler, S. & Kronberger, G. (2013), Visualization of genetic lineages and inheritance information in genetic programming, *in* 'Proceedings of the 15th annual conference companion on Genetic and evolutionary computation', pp. 1351–1358.

Camacho-Villalón, C. L., Dorigo, M. & Stützle, T. (2022*a*), 'An analysis of why cuckoo search does not bring any novel ideas to optimization', *Computers & Operations Research* **142**, 105747.

Camacho-Villalón, C. L., Dorigo, M. & Stützle, T. (2022*b*), 'Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: six misleading optimization techniques inspired by bestial metaphors', *International Transactions in Operational Research* .

Cambria, E., Li, Y., Xing, F. Z., Poria, S. & Kwok, K. (2020), Senticnet 6: Ensemble application of symbolic and subsymbolic ai for sentiment analysis, *in* 'Proceedings of the 29th ACM international conference on information & knowledge management', pp. 105–114.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M. & Elhadad, N. (2015), Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission, *in* 'Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining', pp. 1721–1730.

Castelvecchi, D. (2016), 'Can we open the black box of AI?', *Nature News* **538**(7623), 20.

Chakuma, B. & Helbig, M. (2018), Visualizing the optimization process for multi-objective optimization problems, *in* 'International Conference on Artificial Intelligence and Soft Computing', Springer, pp. 333–344.

Chattopadhay, A., Sarkar, A., Howlader, P. & Balasubramanian, V. N. (2018), Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks, *in* '2018 IEEE winter conference on applications of computer vision (WACV)', IEEE, pp. 839–847.

Chatzimparmpas, A., Martins, R. M., Kucher, K. & Kerren, A. (2021), Visevol: Visual analytics to support hyperparameter search through evolutionary optimization, *in* 'Computer Graphics Forum', Vol. 40, Wiley Online Library, pp. 201–214.

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002), 'SMOTE: Synthetic minority over-sampling technique', *Journal of artificial intelligence research* **16**, 321–357.

Chen, T. & Guestrin, C. (2016), Xgboost: A scalable tree boosting system, *in* 'Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining', pp. 785–794.

Cheng, R., Li, M., Tian, Y., Xiang, X., Zhang, X., Yang, S., Jin, Y. & Yao, X. (2018), Benchmark functions for the cec'2018 competition on many-objective optimization, Technical report.

Chernoff, H. (1973), 'The use of faces to represent points in k-dimensional space graphically', *Journal of the American statistical Association* **68**(342), 361–368.

Chicano, F., Daolio, F., Ochoa, G., Vérel, S., Tomassini, M. & Alba, E. (2012), Local optima networks, landscape autocorrelation and heuristic search performance, *in* 'International Conference on Parallel Problem Solving from Nature', Springer, pp. 337–347.

Chiu, P. & Bloebaum, C. L. (2010), 'Hyper-radial visualization (HRV) method with range-based preferences for multi-objective decision making', *Structural and Multidisciplinary Optimization* **40**(1-6), 97.

Cleveland, W. S. & McGill, R. (1984), 'The many faces of a scatterplot', *Journal of the American Statistical Association* **79**(388), 807–822.

Confalonieri, R., Coba, L., Wagner, B. & Besold, T. R. (2021), 'A historical perspective of explainable artificial intelligence', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **11**(1), e1391.

Corne, D. W. & Knowles, J. D. (2007), Techniques for highly multiobjective optimisation: some nondominated points are better than others, *in* 'Proceedings of the 9th annual conference on Genetic and evolutionary computation', pp. 773–780.

Craven, M. J. & Jimbo, H. C. (2014), EA stability visualization: perturbations, metrics and performance, *in* 'Proceedings of the Companion Publication of GECCO 2014', pp. 1083–1090.

Črepinšek, M., Liu, S.-H. & Mernik, M. (2013), 'Exploration and exploitation in evolutionary algorithms: A survey', *ACM computing surveys (CSUR)* **45**(3), 1–33.

Črepinšek, M., Mernik, M. & Liu, S.-H. (2011), 'Analysis of exploration and exploitation in evolutionary algorithms by ancestry trees', *International Journal of Innovative Computing and Applications* **3**(1), 11–19.

Cruz, A., Machado, P., Assunção, F. & Leitão, A. (2015), Elicit: Evolutionary computation visualization, *in* 'Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation', pp. 949–956.

Daida, J. M., Ward, D. J., Hilss, A. M., Long, S. L., Hodges, M. R. & Kriesel, J. T. (2004), Visualizing the loss of diversity in genetic programming, *in* 'Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)', Vol. 2, IEEE, pp. 1225–1232.

Dandl, S., Molnar, C., Binder, M. & Bischl, B. (2020), Multi-objective counterfactual explanations, *in* 'International Conference on Parallel Problem Solving from Nature', Springer, pp. 448–469.

De Lorenzo, A., Medvet, E., Tušar, T. & Bartoli, A. (2019), An analysis of dimensionality reduction techniques for visualizing evolution, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', pp. 1864–1872.

De Silva, V. & Tenenbaum, J. B. (2004), Sparse multidimensional scaling using landmark points, Technical report, technical report, Stanford University.

Deb, K., Agrawal, R. B. et al. (1995), 'Simulated binary crossover for continuous search space', *Complex systems* **9**(2), 115–148.

Deb, K. & Beyer, H.-G. (2001), 'Self-adaptive genetic algorithms with simulated binary crossover', *Evolutionary computation* **9**(2), 197–221.

Deb, K., Goyal, M. et al. (1996), 'A combined genetic adaptive search (geneas) for engineering design', *Computer Science and informatics* **26**, 30–45.

Deb, K. & Jain, H. (2013), 'An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints', *IEEE transactions on evolutionary computation* **18**(4), 577–601.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002), 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *IEEE transactions on evolutionary computation* **6**(2), 182–197.

Deb, K., Thiele, L., Laumanns, M. & Zitzler, E. (2002), Scalable multi-objective optimization test problems, *in* 'Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)', Vol. 1, IEEE, pp. 825–830.

Dhiman, G. & Kaur, A. (2019), 'Hkn-rvea: a novel many-objective evolutionary algorithm for car side impact bar crashworthiness problem', *International journal of vehicle design* **80**(2-4), 257–284.

Eiben, A. E., Smith, J. E. et al. (2003), *Introduction to evolutionary computing*, Vol. 53, Springer.

Eisen, M. B., Spellman, P. T., Brown, P. O. & Botstein, D. (1998), 'Cluster analysis and display of genome-wide expression patterns', *Proceedings of the National Academy of Sciences* **95**(25), 14863–14868.

Eriksen, C. & Hauri, A. (2022), 'When crises collide: Energy, security, climate change', *CSS Policy Perspectives* **10**(8).

Everson, R. M. & Fieldsend, J. E. (2006), 'Multi-class ROC analysis from a multi-objective optimisation perspective', *Pattern Recognition Letters* **27**(8), 918–927.

Farina, M. & Amato, P. (2002), On the optimal solution definition for many-criteria optimization problems, *in* '2002 annual meeting of the North American fuzzy information processing society proceedings. NAFIPS-FLINT 2002 (Cat. No. 02TH8622)', IEEE, pp. 233–238.

Farina, M. & Amato, P. (2003), Fuzzy optimality and evolutionary multiobjective optimization, *in* 'International Conference on Evolutionary Multi-Criterion Optimization', Springer, pp. 58–72.

Fefferman, C., Mitter, S. & Narayanan, H. (2016), 'Testing the manifold hypothesis', *Journal of the American Mathematical Society* **29**(4), 983–1049.

Fieldsend, J. E., Chugh, T., Allmendinger, R. & Miettinen, K. (2021), 'A visualizable test problem generator for many-objective optimization', *IEEE Transactions on Evolutionary Computation* **26**(1), 1–11.

Filipič, B. & Tušar, T. (2018), A taxonomy of methods for visualizing Pareto front approximations, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', pp. 649–656.

Fleischer, M. (2003), The measure of pareto optima applications to multi-objective metaheuristics, *in* 'International Conference on Evolutionary Multi-Criterion Optimization', Springer, pp. 519–533.

Fonseca, C. M. & Fleming, P. J. (1993), Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization, *in* 'ICGA 93', Vol. 93, Morgan Kaufmann, pp. 416–423.

Fonseca, V. G. d., Fonseca, C. M. & Hall, A. O. (2001), Inferential performance assessment of stochastic optimisers and the attainment function, *in* 'International Conference on Evolutionary Multi-Criterion Optimization', Springer, pp. 213–225.

Friedman, J. H. (2001), 'Greedy function approximation: a gradient boosting machine', *Annals of statistics* pp. 1189–1232.

Fyvie, M., McCall, J. A. & Christie, L. A. (2021), Towards explainable metaheuristics: Pca for trajectory mining in evolutionary algorithms, *in* 'International Conference on Innovative Techniques and Applications of Artificial Intelligence', Springer, pp. 89–102.

Gao, H., Nie, H. & Li, K. (2019), Visualisation of Pareto front approximation: A short survey and empirical comparisons, *in* '2019 IEEE Congress on Evolutionary Computation (CEC)', IEEE, pp. 1750–1757.

García-Vico, Á. M., González, P., Carmona, C. J. & del Jesus, M. J. (2019), 'Study on the use of different quality measures within a multi-objective evolutionary algorithm approach for emerging pattern mining in big data environments', *Big Data Analytics* **4**(1), 1–15.

Géron, A. (2019), *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*, "O'Reilly Media, Inc.".

Goldberg, D. E., Lingle, R. et al. (1985), Alleles, loci, and the traveling salesman problem, *in* 'Proceedings of an international conference on genetic algorithms and their applications', Vol. 154, Lawrence Erlbaum Hillsdale, NJ, pp. 154–159.

Goldstein, A., Kapelner, A., Bleich, J. & Pitkin, E. (2015), 'Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation', *journal of Computational and Graphical Statistics* **24**(1), 44–65.

Golub, G. H. & Van Loan, C. F. (1996), 'Matrix computations', *Johns Hopkins Universtiy Press, 3rd edtion* .

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014), 'Generative adversarial nets', *Advances in neural information processing systems* **27**.

Goodman, B. & Flaxman, S. (2017), 'European union regulations on algorithmic decision-making and a "right to explanation"', *AI magazine* **38**(3), 50–57.

Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F. & Pedreschi, D. (2018), 'A survey of methods for explaining black box models', *ACM computing surveys (CSUR)* **51**(5), 1–42.

Hart, E. & Ross, P. (2001), 'Gavel-a new tool for genetic algorithm visualization', *IEEE Transactions on Evolutionary Computation* **5**(4), 335–348.

Hastie, T. J. (2017), Generalized additive models, *in* 'Statistical models in S', Routledge, pp. 249–307.

He, H., Bai, Y., Garcia, E. A. & Li, S. (2008), Adasyn: Adaptive synthetic sampling approach for imbalanced learning, *in* '2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)', IEEE, pp. 1322–1328.

He, Z. & Yen, G. G. (2015), 'Visualization and performance metric in many-objective optimization', *IEEE Transactions on Evolutionary Computation* **20**(3), 386–402.

Hettenhausen, J., Lewis, A. & Mostaghim, S. (2010), 'Interactive multi-objective particle swarm optimization with heatmap-visualization-based user interface', *Engineering Optimization* **42**(2), 119–139.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Hoffman, P., Grinstein, G., Marx, K., Grosse, I. & Stanley, E. (1997), DNA visual and analytic data mining, *in* 'Proceedings. Visualization'97 (Cat. No. 97CB36155)', IEEE, pp. 437–441.

Hoffman, R. R., Mueller, S. T., Klein, G. & Litman, J. (2018), 'Metrics for explainable AI: Challenges and prospects', *arXiv preprint arXiv:1812.04608* .

Holland, J. H. (1975), *Adaptation in natural and artificial systems*, University of Michigan Press.

Hornby, G. S. & Yu, T. (2007), 'Ec practitioners: Results of the first survey', *ACM SIGEVOlution* **2**(1), 2–8.

Huband, S., Barone, L., While, L. & Hingston, P. (2005), A scalable multi-objective test problem toolkit, *in* 'International Conference on Evolutionary Multi-Criterion Optimization', Springer, pp. 280–295.

Huband, S., Hingston, P., Barone, L. & While, L. (2006), 'A review of multiobjective test problems and a scalable test problem toolkit', *IEEE Transactions on Evolutionary Computation* **10**(5), 477–506.

Ibrahim, A., Rahnamayan, S., Martin, M. V. & Deb, K. (2016), 3D-RadVis: Visualization of Pareto front in many-objective optimization, *in* '2016 IEEE Congress on Evolutionary Computation (CEC)', IEEE, pp. 736–745.

Inselberg, A. & Dimsdale, B. (1990), Parallel coordinates: a tool for visualizing multi-dimensional geometry, *in* 'Proceedings of the First IEEE Conference on Visualization: Visualization90', IEEE, pp. 361–378.

Ishibuchi, H., Imada, R., Setoguchi, Y. & Nojima, Y. (2016), Performance comparison of NSGA-II and NSGA-III on various many-objective test problems, *in* '2016 IEEE Congress on Evolutionary Computation (CEC)', IEEE, pp. 3045–3052.

Ishibuchi, H., Yoshida, T. & Murata, T. (2003), 'Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling', *IEEE transactions on evolutionary computation* **7**(2), 204–223.

Jerebic, J., Mernik, M., Liu, S.-H., Ravber, M., Baketarić, M., Mernik, L. & Črepinšek, M. (2021), 'A novel direct measure of exploration and exploitation based on attraction basins', *Expert Systems with Applications* **167**, 114353.

Jiang, S., Zou, J., Yang, S. & Yao, X. (2022), 'Evolutionary dynamic multi-objective optimisation: A survey', *ACM Computing Surveys (CSUR)* .

Jocher, G., Stoken, A., Borovec, J., NanoCode012, ChristopherSTAN, Changyu, L., Laughing, tkianai, yxNONG, Hogan, A., lorenzomammana, AlexWang1900, Chaurasia, A., Diaconu, L., Marc, wanghaoyang0106, ml5ah, Doug, Durgesh, Ingham, F., Frederik, Guilhen, Colmagro, A., Ye, H., Jacobsolawetz, Poznanski, J., Fang, J., Kim, J., Doan, K. & Yu, L. (2021), 'ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration'.
URL: *https://doi.org/10.5281/zenodo.4418161*

John, H. (1976), 'Holland. 1976. adaptation', *Progress in theoretical biology* **4**, 263–293.

Johnson, B. & Shneiderman, B. (1998), Tree-maps: A space filling approach to the visualization of hierarchical information structures, Technical report.

Kandogan, E. (2000), Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions, *in* 'Proceedings of the IEEE information visualization symposium', Vol. 650, Citeseer, p. 22.

Kerren, A. & Egger, T. (2005), Eavis: A visualization tool for evolutionary algorithms, *in* '2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)', IEEE, pp. 299–301.

Kerschke, P., Hoos, H. H., Neumann, F. & Trautmann, H. (2019), 'Automated algorithm selection: Survey and perspectives', *Evolutionary computation* **27**(1), 3–45.

Keshk, M. (2017), Semantic evolutionary visualization, *in* 'International Conference on Swarm Intelligence', Springer, pp. 624–635.

Kim, B., Glassman, E., Johnson, B. & Shah, J. (2015), 'ibcm: Interactive bayesian case model empowering humans via intuitive interaction'.

Kitamura, T. & Fukunaga, A. (2021), Visualizing fitnesses and constraint violations in single-objective optimization, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', pp. 1953–1960.

Konig, R., Johansson, U. & Niklasson, L. (2008), G-rex: A versatile framework for evolutionary data mining, *in* '2008 IEEE International Conference on Data Mining Workshops', IEEE, pp. 971–974.

Köppen, M. & Yoshida, K. (2007), Substitute distance assignments in NSGA-II for handling many-objective optimization problems, *in* 'International Conference on Evolutionary Multi-Criterion Optimization', Springer, pp. 727–741.

Koza, J. R. et al. (1989), Hierarchical genetic algorithms operating on populations of computer programs., *in* 'IJCAI', Vol. 89, pp. 768–774.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), 'Imagenet classification with deep convolutional neural networks', *Advances in neural information processing systems* **25**.

Laugel, T., Lesot, M.-J., Marsala, C., Renard, X. & Detyniecki, M. (2017), 'Inverse classification for comparison-based interpretability in machine learning', *arXiv preprint arXiv:1712.08443* .

Lavinas, Y., Aranha, C. & Ochoa, G. (2022), Search trajectories networks of multiobjective evolutionary algorithms, *in* 'International Conference on the Applications of Evolutionary Computation (Part of EvoStar)', Springer, pp. 223–238.

LeCun, Y., Haffner, P., Bottou, L. & Bengio, Y. (1999), Object recognition with gradient-based learning, *in* 'Shape, contour and grouping in computer vision', Springer, pp. 319–345.

Li, E., Xia, C., Zhao, D., Lu, L., Xiang, J., He, Y., Wang, J. & Wu, J. (2018), Improved heatmap visualization of Pareto-optimal set in multi-objective optimization of defensive strategy, *in* '2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)', IEEE, pp. 345–352.

Lipton, Z. C. (2018), 'The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.', *Queue* **16**(3), 31–57.

López-Ibáñez, M., Paquete, L. & Stützle, T. (2010), Exploratory analysis of stochastic local search algorithms in biobjective optimization, *in* 'Experimental methods for the analysis of optimization algorithms', Springer, pp. 209–222.

Lowe, D. & Tipping, M. E. (1997), Neuroscale: Novel topographic feature extraction using RBF networks, *in* 'Advances in neural information processing systems', pp. 543–549.

Lundberg, S. M., Erion, G. G. & Lee, S.-I. (2018), 'Consistent individualized feature attribution for tree ensembles', *arXiv preprint arXiv:1802.03888* .

Lundberg, S. M. & Lee, S.-I. (2017), 'A unified approach to interpreting model predictions', *Advances in neural information processing systems* **30**.

Luo, W., Lin, X., Li, C., Yang, S. & Shi, Y. (2022), 'Benchmark functions for cec 2022 competition on seeking multiple optima in dynamic environments', *arXiv preprint arXiv:2201.00523* .

Manavi, F. & Hamzeh, A. (2019), A new approach for malware detection based on evolutionary algorithm, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', pp. 1619–1624.

Masafumi, Y., Tomohiro, Y. & Takeshi, F. (2010), Study on effect of MOGA with interactive island model using visualization, *in* 'IEEE Congress on Evolutionary Computation', IEEE, pp. 1–6.

McInnes, L., Healy, J. & Melville, J. (2018), 'Umap: Uniform manifold approximation and projection for dimension reduction', *arXiv preprint arXiv:1802.03426* .

McPhee, N. F., Casale, M. M., Finzel, M., Helmuth, T. & Spector, L. (2016), Visualizing genetic programming ancestries, *in* 'Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion', pp. 1419–1426.

McPhee, N. F., Casale, M. M., Finzel, M., Helmuth, T. & Spector, L. (2017), Visualizing genetic programming ancestries using graph databases, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', pp. 245–246.

McPhee, N. F., Finzel, M. D., Casale, M. M., Helmuth, T. & Spector, L. (2018), A detailed analysis of a pushgp run, *in* 'Genetic Programming Theory and Practice XIV', Springer, pp. 65–83.

Medvet, E., Virgolin, M., Castelli, M., Bosman, P. A., Gonçalves, I. & Tušar, T. (2018), 'Unveiling evolutionary algorithm representation with du maps', *Genetic Programming and Evolvable Machines* **19**, 351–389.

Miettinen, K. (2012), *Nonlinear multiobjective optimization*, Vol. 12, Springer Science & Business Media.

Minsky, M. & Papert, S. (1969), 'An introduction to computational geometry', *Cambridge tiass., HIT* **479**, 480.

Mosetti, G., Poloni, C. & Diviacco, B. (1994), 'Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm', *Journal of Wind Engineering and Industrial Aerodynamics* **51**(1), 105–116.

Mouret, J.-B. & Clune, J. (2015), 'Illuminating search spaces by mapping elites', *arXiv preprint arXiv:1504.04909* .

Nelder, J. A. & Wedderburn, R. W. (1972), 'Generalized linear models', *Journal of the Royal Statistical Society: Series A (General)* **135**(3), 370–384.

Nikolos, I. K., Valavanis, K. P., Tsourveloudis, N. C. & Kostaras, A. N. (2003), 'Evolutionary algorithm based offline/online path planner for uav navigation', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **33**(6), 898–912.

Obayashi, S. & Sasaki, D. (2003), Visualization and data mining of Pareto solutions using self-organizing map, *in* 'International Conference on Evolutionary Multi-Criterion Optimization', Springer, pp. 796–809.

Ochoa, G., Malan, K. M. & Blum, C. (2020), Search trajectory networks of population-based algorithms in continuous spaces, *in* 'International Conference on the Applications of Evolutionary Computation (Part of EvoStar)', Springer, pp. 70–85.

Ochoa, G., Malan, K. M. & Blum, C. (2021), 'Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics', *Applied Soft Computing* **109**, 107492.

Ochoa, G., Tomassini, M., Vérel, S. & Darabos, C. (2008), A study of NK landscapes' basins and local optima networks, *in* 'Proceedings of the 10th annual conference on Genetic and evolutionary computation', pp. 555–562.

Ochoa, G., Verel, S., Daolio, F. & Tomassini, M. (2014), Local optima networks: A new model of combinatorial fitness landscapes, *in* 'Recent advances in the theory and application of fitness landscapes', Springer, pp. 233–262.

O'Leary, D. E. (2019), 'Google's duplex: Pretending to be human', *Intelligent Systems in Accounting, Finance and Management* **26**(1), 46–53.

Phillips, P. J., Hahn, C. A., Fontana, P. C., Broniatowski, D. A. & Przybocki, M. A. (2020), 'Four principles of explainable artificial intelligence', *Gaithersburg, Maryland* .

Pohlheim, H. (1999), Visualization of evolutionary algorithms-set of standard techniques and multidimensional visualization, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference', Vol. 1, San Francisco, CA., pp. 533–540.

Pryke, A., Mostaghim, S. & Nazemi, A. (2007), Heatmap visualization of population based multi objective algorithms, *in* 'International Conference on Evolutionary Multi-Criterion Optimization', Springer, pp. 361–375.

Pu, P. & Chen, L. (2006), Trust building with explanation interfaces, *in* 'Proceedings of the 11th international conference on Intelligent user interfaces', pp. 93–100.

Rechenberg, I. (1973), 'Evolutionsstrategie', *Optimierung technischer Systeme nach Prinzipien derbiologischen Evolution* .

Ribeiro, M. T., Singh, S. & Guestrin, C. (2016), " 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', pp. 1135–1144.

Riquelme, N., Von Lücken, C. & Baran, B. (2015), Performance metrics in multi-objective optimization, *in* '2015 Latin American computing conference (CLEI)', IEEE, pp. 1–11.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986), 'Learning representations by back-propagating errors', *nature* **323**(6088), 533–536.

Sammon, J. W. (1969), 'A nonlinear mapping for data structure analysis', *IEEE Transactions on computers* **100**(5), 401–409.

Sanchis, J., Martínez, M. A. & Blasco, X. (2008), 'Integrated multiobjective optimization and a priori preferences using genetic algorithms', *Information Sciences* **178**(4), 931–951.

Schäpermeier, L., Grimme, C. & Kerschke, P. (2020), One PLOT to show them all: Visualization of efficient sets in multi-objective landscapes, *in* 'International Conference on Parallel Problem Solving from Nature', Springer, pp. 154–167.

Sekanina, L. & Kapusta, V. (2016), Visualisation and analysis of genetic records produced by cartesian genetic programming, *in* 'Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion', pp. 1411–1418.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D. & Batra, D. (2017), Grad-cam: Visual explanations from deep networks via gradient-based localization, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 618–626.

Shaik, J. S. & Yeasin, M. (2006), Visualization of high dimensional data using an automated 3D star co-ordinate system, *in* 'The 2006 IEEE International Joint Conference on Neural Network Proceedings', IEEE, pp. 1339–1346.

Shrikumar, A., Greenside, P. & Kundaje, A. (2017), Learning important features through propagating activation differences, *in* 'International conference on machine learning', PMLR, pp. 3145–3153.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. et al. (2016), 'Mastering the game of go with deep neural networks and tree search', *nature* **529**(7587), 484–489.

Simonyan, K., Vedaldi, A. & Zisserman, A. (2013), 'Deep inside convolutional networks: Visualising image classification models and saliency maps', *arXiv preprint arXiv:1312.6034* .

Singh, M., Brownlee, A. E. & Cairns, D. (2022), Towards explainable metaheuristic: mining surrogate fitness models for importance of variables, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', pp. 1785–1793.

Smilkov, D., Thorat, N., Kim, B., Viégas, F. & Wattenberg, M. (2017), 'Smoothgrad: removing noise by adding noise', *arXiv preprint arXiv:1706.03825* .

Smith-Miles, K. A. (2009), 'Cross-disciplinary perspectives on meta-learning for algorithm selection', *ACM Computing Surveys (CSUR)* **41**(1), 1–25.

Sörensen, K. (2015), 'Metaheuristics—the metaphor exposed', *International Transactions in Operational Research* **22**(1), 3–18.

Springenberg, J. T., Dosovitskiy, A., Brox, T. & Riedmiller, M. (2014), 'Striving for simplicity: The all convolutional net', *arXiv preprint arXiv:1412.6806* .

Storn, R. & Price, K. (1997), 'Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces', *Journal of global optimization* **11**(4), 341–359.

Su, G., Wei, D., Varshney, K. R. & Malioutov, D. M. (2015), 'Interpretable two-level boolean rule learning for classification', *arXiv preprint arXiv:1511.07361* .

Sundararajan, M., Taly, A. & Yan, Q. (2017), Axiomatic attribution for deep networks, *in* 'International conference on machine learning', PMLR, pp. 3319–3328.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. (2015), Going deeper with convolutions, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1–9.

Talukder, A. K. A. & Deb, K. (2020), Paletteviz with star-coordinates: An improved method for high-dimensional pareto-optimal front visualization and decision-making, *in* '2020 IEEE Symposium Series on Computational Intelligence (SSCI)', IEEE, pp. 2186–2193.

Targosz, M., Szumowski, M., Skarka, W. & Przystałka, P. (2013), Velocity planning of an electric vehicle using an evolutionary algorithm, *in* 'International Conference on Transport Systems Telematics', Springer, pp. 171–177.

Tenenbaum, J. B., De Silva, V. & Langford, J. C. (2000), 'A global geometric framework for nonlinear dimensionality reduction', *science* **290**(5500), 2319–2323.

Ting, C.-K., Su, C.-H. & Lee, C.-N. (2010), 'Multi-parent extension of partially mapped crossover for combinatorial optimization problems', *Expert Systems with Applications* **37**(3), 1879–1886.

Tiwari, A., Hoyos, P. N., Hutabarat, W., Turner, C., Ince, N., Gan, X.-P. & Prajapat, N. (2015), 'Survey on the use of computational optimisation in uk engineering companies', *CIRP Journal of Manufacturing Science and Technology* **9**, 57–68.

Tiwari, A., Roy, R., Jared, G. & Munaux, O. (2002), 'Evolutionary-based techniques for real-life optimisation: development and testing', *Applied Soft Computing* **1**(4), 301–329.

Tušar, T., Brockhoff, D., Hansen, N. & Auger, A. (2016), 'Coco: the bi-objective black box optimization benchmarking (bbob-biobj) test suite', *ArXiv e-prints* .

Tušar, T. & Filipič, B. (2014*a*), 'Visualization of Pareto front approximations in evolutionary multiobjective optimization: A critical review and the prosection method', *IEEE Transactions on Evolutionary Computation* **19**(2), 225–245.

Tušar, T. & Filipič, B. (2014*b*), 'Visualizing exact and approximated 3d empirical attainment functions', *Mathematical Problems in Engineering* **2014**.

Valdés, J. J. & Barton, A. J. (2007), Visualizing high dimensional objective spaces for multi-objective optimization: A virtual reality approach, *in* '2007 IEEE Congress on Evolutionary Computation', IEEE, pp. 4199–4206.

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W. & Kavukcuoglu, K. (2016), 'WaveNet: A generative model for raw audio', *SSW* **125**, 2.

van der Blom, K., Deist, T. M., Tušar, T., Marchi, M., Nojima, Y., Oyama, A., Volz, V. & Naujoks, B. (2020), Towards realistic optimization benchmarks: a questionnaire on the properties of real-world problems, *in* 'Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion', pp. 293–294.

Van der Maaten, L. & Hinton, G. (2008), 'Visualizing data using t-sne.', *Journal of machine learning research* **9**(11).

Van Veldhuizen, D. A. & Lamont, G. B. (1998), Multiobjective evolutionary algorithm research: A history and analysis, Technical report, Citeseer.

Varga, M. N., Gaudl, S. E. & Walker, D. J. (2021), Many-objective population visualisation with geons, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', pp. 1961–1969.

Verel, S., Daolio, F., Ochoa, G. & Tomassini, M. (2011), Local optima networks with escape edges, *in* 'International Conference on Artificial Evolution (Evolution Artificielle)', Springer, pp. 49–60.

Verel, S., Ochoa, G. & Tomassini, M. (2010), 'Local optima networks of NK landscapes with neutrality', *IEEE Transactions on Evolutionary Computation* **15**(6), 783–797.

Vilone, G. & Longo, L. (2020), 'Explainable artificial intelligence: a systematic review', *arXiv preprint arXiv:2006.00093* .

Vincalek, J., Walton, S. & Evans, B. (2021), It's the journey not the destination: building genetic algorithms practitioners can trust, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', pp. 231–232.

Wachter, S., Mittelstadt, B. & Russell, C. (2017), 'Counterfactual explanations without opening the black box: Automated decisions and the GDPR', *Harv. JL & Tech.* **31**, 841.

Walker, D. & Craven, M. J. (2018*a*), 'Visualising the operation of evolutionary algorithms optimising water distribution network design problems'.

Walker, D., Fieldsend, J. & Everson, R. (2012), Visualising many-objective populations, *in* 'Proceedings of the 14th annual conference companion on Genetic and evolutionary computation', pp. 451–458.

Walker, D. J. (2012), 'Ordering and visualisation of many-objective populations'.

Walker, D. J. (2015), Visualising multi-objective populations with treemaps, *in* 'Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation', pp. 963–970.

Walker, D. J. & Craven, M. J. (2018*b*), Toward the online visualisation of algorithm performance for parameter selection, *in* 'International Conference on the Applications of Evolutionary Computation', Springer, pp. 547–560.

Walker, D. J., Everson, R. M. & Fieldsend, J. E. (2010), Visualisation and ordering of many-objective populations, *in* 'IEEE Congress on Evolutionary Computation', IEEE, pp. 1–8.

Walker, D. J., Everson, R. M. & Fieldsend, J. E. (2012), 'Visualizing mutually nondominating solution sets in many-objective optimization', *IEEE Transactions on Evolutionary Computation* **17**(2), 165–184.

Walton, S. P. (2013), *Gradient free optimisation in selected engineering applications*, Swansea University (United Kingdom).

Wang, Q., Guan, M., Huang, W., Wang, L., Wang, Z., Liu, S. & Savić, D. (2021), 'Visualisation of the combinatorial effects within evolutionary algorithms: the compass plot', *Journal of Hydroinformatics* **23**(3), 517–528.

Witowski, K., Liebscher, M. & Goel, T. (2009), Decision making in multi-objective optimization for industrial applications–data mining and visualization of pareto data, *in* 'Proceedings of the 7th European LS-DYNA Conference, Salzburg, Austria'.

Wu, A. S., De Jong, K. A., Burke, D. S., Grefenstette, J. J. & Ramsey, C. L. (1999), Visual analysis of evolutionary algorithms, *in* 'Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)', Vol. 2, IEEE, pp. 1419–1425.

Yao, X., Liu, Y. & Lin, G. (1999), 'Evolutionary programming made faster', *IEEE Transactions on Evolutionary computation* **3**(2), 82–102.

Zhang, Q. & Li, H. (2007), 'MOEA/D: A multiobjective evolutionary algorithm based on decomposition', *IEEE Transactions on evolutionary computation* **11**(6), 712–731.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A. & Torralba, A. (2016), Learning deep features for discriminative localization, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2921–2929.

Zhu, J., Liapis, A., Risi, S., Bidarra, R. & Youngblood, G. M. (2018), Explainable AI for designers: A human-centered perspective on mixed-initiative co-creation, *in* '2018 IEEE Conference on Computational Intelligence and Games (CIG)', IEEE, pp. 1–8.

Zhu, Q., Oganov, A. R., Glass, C. W. & Stokes, H. T. (2012), 'Constrained evolutionary algorithm for structure prediction of molecular crystals: methodology and applications', *Acta Crystallographica Section B: Structural Science* **68**(3), 215–226.

Zitzler, E., Deb, K. & Thiele, L. (2000), 'Comparison of multiobjective evolutionary algorithms: Empirical results', *Evolutionary computation* **8**(2), 173–195.

Zitzler, E., Laumanns, M. & Thiele, L. (2001), 'SPEA2: Improving the strength pareto evolutionary algorithm', *TIK-report* **103**.

Zitzler, E. & Thiele, L. (1999), 'Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach', *IEEE transactions on Evolutionary Computation* **3**(4), 257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M. & Da Fonseca, V. G. (2003), 'Performance assessment of multiobjective optimizers: An analysis and review', *IEEE Transactions on evolutionary computation* **7**(2), 117–132.