Towards explainable metaheuristics: feature mining of search trajectories through principal component projection.

FYVIE, M., MCCALL, J.A.W. and CHRISTIE, L.A.

2025

Towards Explainable Metaheuristics: Feature Mining of Search Trajectories Through Principal Component Projection

MARTIN FYVIE*, JOHN A. W. MCCALL*, and LEE A. CHRISTIE*, Robert Gordon University, Scotland

While population-based metaheuristics have proven useful for refining and improving explainable AI systems, they are seldom the focus of explanatory approaches themselves. This stems from their inherently stochastic, population-driven searches, which complicate the use of standard explainability techniques. In this paper, we present a method to identify which decision variables have the greatest impact during an algorithm's trajectory from random initialization to convergence. We apply Principal Component Analysis to project each population onto a lower-dimensional space, then introduce two metrics—Mean Variable Contribution and Proportion of Aligned Variables—to identify the variables most responsible for guiding the search. Using four different population-based methods (Particle Swarm Optimisation, Genetic Algorithm, Differential Evolution, and Covariance Matrix Adaptation Evolution Strategy) on 24 BBOB benchmark functions in 10 dimensions, we find that these metrics highlight meaningful variable relationships and provide a window into each method's search dynamics. By comparing the features extracted across algorithms and problems, we illustrate how certain variable subsets consistently drive major improvements in solution quality. In doing so, new evolutionary algorithm variants can be designed to take advantage of these influential variables, while also identifying underutilised variables that may benefit alternative search strategies.

CCS Concepts: • **Computing methodologies** → **Search methodologies**; **Genetic algorithms**; • **Human-centred computing** → **Visualisation application domains**.

Additional Key Words and Phrases: Evolutionary Algorithms, Principal Component Analysis, Algorithm Trajectories, Visualisation, Population Diversity

1 INTRODUCTION

As the adoption of Evolutionary Algorithms (EAs) and similar search metaheuristics in applications involving end-user cooperation grows, so too does the need to promote user trust and acceptance of EAs. Explainable Artificial Intelligence (XAI) as a field has, in recent years, seen a large growth in interest driven by this growing adoption.

Recent surveys of the XAI research landscape (Barredo Arrieta et al. 2020; Dwivedi et al. 2023) highlight the broad scope that XAI as a term encompasses. EAs such as Genetic Algorithms (GAs) are often employed in XAI as tools to enhance other methods of explanation generation. Examples found in these surveys include the use of EAs as a tool for fuzzy rule refinement (Duygu Arbatli and Levent Akin 1997) and counterfactual generation (Sharma et al. 2020). Additionally, the integration of evolutionary fuzzy systems in XAI, as discussed in (Fernández et al. 2019), emphasises the significance of evolutionary algorithms in designing interpretable fuzzy systems, helping bridge the gap between complex computational processes and human-centric explanations. It is however noticeable that there has been an under-representation of EAs themselves as the focus of XAI.

*All authors contributed equally to this research.

Authors' Contact Information: Martin Fyvie, m.fyvie@rgu.ac.uk; John A. W. McCall, j.mccall@rgu.ac.uk; Lee A. Christie, l.a.christie@rgu.ac.uk, Robert Gordon University, Aberdeen, Scotland.

A notable common behaviour of EAs is the generation of successive populations of candidate solutions, utilising non-deterministic internal mechanisms. A classic example is the use of selection, crossover, and mutation operators in a GA. This process produces a sequence of populations, a search trajectory, representing where in the problem landscape the algorithm has searched for increasingly improving solutions. The search trajectory is a record of all visited solutions during optimisation with respect to their fitness function value. Each population within the trajectory is a sample of solutions representing the EA's implicit current model of the fitness function. Over time, ideally, these populations will converge on the ideal or near-ideal set of solutions for that given problem. Commonly, the quality of the generated solutions also tends to increase across successive generations and so the search trajectory naturally associates changes in search space variables with fitness improvement.

Principal Component Analysis (PCA) has, for some time, been a popular technique in the realm of data analysis and dimensionality reduction and is the basis for a significant number of dimension reduction methods (Nanga et al. 2021). Its ability to transform high-dimensional data into a lower-dimensional form while preserving the most significant patterns in the data makes it a valuable tool for various applications. In this paper we propose the application of PCA to the analysis of search trajectories. Our hypothesis is that, by reducing the dimensionality of search trajectories, PCA can highlight the most influential components or features of the search space that drive the behaviour of an algorithm. This distilled representation can then be used to provide more interpretable and concise explanations of the algorithm's behaviour, aiding in the understanding and *trustworthiness* (Jolliffe and Cadima 2016) of the EA .

Contribution

In this paper, we evaluate our hypotheses that the search trajectories generated by a representative collection of EAs can be mined for explanatory features relating search space variables and fitness. The features generated have some power to explain the behaviour of the algorithm as it seeks better-quality solutions. We generate a series of algorithm search trajectories by solving a set of "black-box" optimisation benchmark problems using Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Differential Evolution (DE), a Genetic Algorithm (GA) and a Particle Swarm Optimiser (PSO). The resulting search trajectories are then analysed for explanatory features we aim to relate to algorithm search behaviour by highlighting variables of significant influence on the search paths taken.

To achieve this, two metrics of variable importance are applied to the search trajectories generated by the collection of EAs. These metrics, defined in Section 4, measure the mean variable contribution and variable alignment to fitness change. The first metric aims to detect the mean variable influence across multiple optimisation runs in relation to a specific component. The second measures the proportional variable alignment to fitness. These metrics are designed to identify which variable or component, at any stage or window of generations of a search trajectory, is dominant in terms of contribution to fitness gains during that period. By measuring the dominance of the components, we show that we can relate geometric features derived from search trajectories to algorithm search behaviour. We consider this analogous to the concept of causality in machine learning, where, as noted in (Barredo Arrieta et al. 2020), while causality and the inference of variable relationships require background knowledge of the problem, explainable ML models that detect such interactions can support existing findings, "…to provide a first intuition of possible causal relationships within the available data …".

By simplifying the explanation representation into sets of variable relationships, we can provide more interpretable and concise explanations of the algorithm's behaviour, aiding in the understanding and trustworthiness of the AI system (Jolliffe and Cadima 2016). This representation can be used to highlight variables with significant contribution for further analysis. It also allows us to differentiate distinct algorithm search behaviours on the same set of problems. This has the potential to ensure that any new EA iteration can exploit the same variables as others to ensure high quality solutions are being found. Our variable alignment results further support this

by providing the ability to compare search behaviours over time with other EAs by observing the relationship between algorithm performance and mined variable importances across multiple EAs.

1.1 Related Work

The exploration of algorithm search trajectories is a highly active area of research. Early examples involving dimension reduction include Sammon mapping (Pohlheim 2006), which has been used to reduce the dimensions of data for the purpose of visualisation, as well as Euclidean Embedding (Michalak 2019), used for similar purposes. These methods are aimed to make the exploration of search trajectories more readily understandable through the reduction of dimensions and visualisation of the trajectories themselves. More recent examples include the work in (Fyvie et al. 2021) which shows how decomposition techniques can be used to link low-level variable interaction detection to a set of features mined from the lower-dimension subspaces of binary encoded optimisation problems. Similarly, this method has been used on discrete integer representations as in (Fyvie et al. 2023).

Landscape analysis has emerged as an important technique in understanding the behaviour and performance of optimisation algorithms. This approach provides insights into the structure and characteristics of the problem landscape, aiding in the design and adaptation of algorithms (Malan and Engelbrecht 2021). The work in (Jankovic 2021) highlights the importance of leveraging landscape analysis for dynamic algorithm selection and configuration. Surrogate modelling (Jin 2011), which extracts feature importances from search trajectories, is done by creating a model of solution populations, capturing fitness function sensitivities to specific variables. This approach mirrors (Singh et al. 2022; Wallace et al. 2021), where a surrogate model identifies feature importance throughout a search trajectory. These features highlight algorithm learning steps and solution variable patterns. Landscape analysis and surrogate modelling can be used together to help explain and improve algorithm performance as shown in (Pitra et al. 2019). This approach also involved the use of algorithm state-variables which provide a detailed representation of the internal workings of an algorithm at any given point in its execution. These variables capture the current state of the algorithm and can be used to understand its behaviour, especially in optimisation and search algorithms. This was used in (Holena 2020) where the internal state variables of a random forest algorithm are explored to provide explanations for anomalies detected by the model.

Work involving the creation of Search Trajectory Networks (STNs) (Ochoa et al. 2021a,b) has been shown to highlight algorithm behavioural differences through their search of the problem landscape. This technique is based on the concept of Local Optima Networks (Adair et al. 2019) in which the fitness landscape can be represented as a set of local optima or basins of attraction. An algorithm's ability to move from one basin to another is mapped in this network. This technique has been used to develop methods of analysis including landscape-aware analysis (Chicano et al. 2023) and multi-objective pareto-front analysis and visualisation (Liefooghe et al. 2023). Further work regarding the differentiation of evolutionary algorithms from generated landscape features, in this case in randomly-generated multi-objective interpolated continuous optimisation problems (MO-ICOPs), can be seen in (Liefooghe et al. 2021). This paper highlights the ability of such features to be used, in combination with a classification model, as predictors of differing convergence behaviour in a selection of evolutionary algorithms. An example of landscape analysis utilising decomposition methods for multi-objective problems can be seen in (Cosson et al. 2021). Here, the work introduces decomposition-based multi-objective landscape features to enhance automated algorithm selection. By decomposing multi-objective problems into single-objective sub-problems, the work extracts informative features that capture global landscape properties, aiding in the prediction of algorithm performance. Network-based research covering some of the same algorithms outlined later in this paper includes the study of both Differential Evolution (Gajdo et al. 2015; Skanderova et al. 2016) and Particle Swarm optimisers (Oliveira et al. 2014; Taw et al. 2019). These methods involve the study of interactions between individual solutions within a population and how this may affect overall search paths. The same methods have been used to attempt to

identify and showcase the differences in behaviour during the theorised "exploration" and "exploitation" phases of a search process between algorithms.

More recently, search trajectories have been explored for the purpose of algorithm selection improvement and problem class identification. The work in (Jankovic et al. 2022) introduces a novel online algorithm selection method, where the authors use initial optimisation runs to generate features that enable a "warm-start" condition, which can aid in algorithm selection based on predicted performance. This is achieved using a combination of exploratory landscape features and internal state variables provided by a CMA-ES iteration. In (Cenikj et al. 2023), the authors propose DynamoRep, a feature extraction method that characterizes optimisation problem instances based on the trajectories of optimisation algorithms. The trajectory features used in this work include population-based descriptive metrics, such as minimum, maximum, and standard deviation values, to capture and describe the interaction between a set of algorithms and specific BBOB problem instances. Our approach differs from these methods by treating the entire population at each generation as the trajectory data, rather than isolated points or summaries. This allows us to capture variable influence across all solutions, enabling a generation-level analysis that reveals how the importance of individual variables evolves over the course of the search. By applying PCA to this population data, we decompose the search trajectory into subspaces that reveal dominant linear relationships, enabling us to assess how particular variables align with higher-quality solutions in different points of interest in the search. This approach provides a structured, geometrically sensitive analysis that directly relates population dynamics back to individual variables, aiming to provide some level of explanation regarding variable interactions and search behaviour across benchmark problems and algorithms.

In this paper we extend the methods used in (Fyvie et al. 2021, 2023) to continuous spaces and adapt the approach to measure "Variable Alignment", a metric derived from the calculated variable contributions produced by PCA. By giving a clearly defined structure to the algorithm trajectory, we can relate changes in overall population fitness to these explanatory features for the purpose of better understanding algorithm behaviour on a large set of benchmark problems. The techniques in this paper build upon that by attempting to relate geometrically sensitive features derived from trajectories to algorithm search behaviour.

1.2 Feature Pipeline

We propose a structured "pipeline" format that represents the common necessary steps to generate a set of explanatory features from the analysis of a search algorithm's output. These steps have been collected and given descriptive terms as seen in Figure 1. The pipeline begins with the step of "Data Acquisition". The acquisition of data in this paper takes the form of generating a large number of optimisation trajectories through the application of a collection of EAs to a set of benchmark problems. The next step, "Population Collection", in the pipeline is the collection of the input data into a form capable of being used in subsequent analysis. This step may include transformations to the data such as dimension reduction or clustering. The "Feature Extraction" step is the point at which the main analysis techniques are applied to the processed data. The aim of this step is to create a set of features that can be used in the next stage of the process. The features are intended to hold some level of explanatory power regarding algorithm behaviour. The "Explanation Generation" step is where the features generated are used to create some level of direct explanation. An example of this may be the conversion of fuzzy-rule sets into sentences through the application of a semantic engine like Natural Language Generation (NLG) techniques.

Each population can be used to generate many features depending on the method of feature extraction used. Each feature is the result of the application of any given analysis technique such that the output is a descriptive set of features that represent the population and has the potential to be used to generate an explanation. As each step is an important facet of the explanation generation approach taken in our experiments, the research paper will be structured such that each step of the pipeline is represented by its own section in this paper.

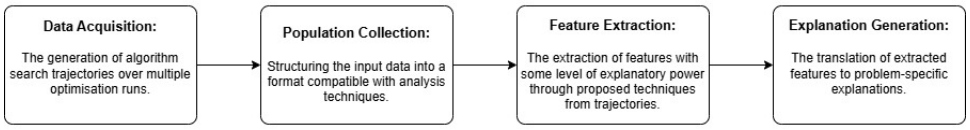


Fig. 1. Initial pipeline steps

1.3 Structure

The rest of the paper is structured as follows. **Section 2** introduces the methods used in the "Data Acquisition" stage of our pipeline. Here we describe the algorithms and problem sets used to generate the search trajectories used in our analysis. **Section 3** introduces the "Population Collection" stage in which we describe the process of applying PCA techniques to create data sets from the trajectory. We justify our decision to use PCA by considering alternatives and show our reasoning. In **Section 4** we outline the "Feature Extraction" processes used to mine the resulting datasets for explanatory features of variable contributions and population alignment. **Section 5** outlines the results of the "Explanation Generation" process and the initial explanations that can be derived from the results. **Section 6** is used to reflect on the methods.

2 DATA ACQUISITION

To generate the required search trajectories for our analysis, a selection of four population-based metaheuristics was used to solve a set of real-valued optimisation problems. This process resulted in the creation of a collection of search trajectories for use in our analysis. The algorithms and problems used are detailed in this section.

2.1 Optimisation Problems

The Black Box Optimisation Benchmarking (BBOB) set of optimisation problems is a well-known and often used problem set for the measurement and comparison of optimisation techniques. Consisting of 24 real-valued problems, the function problem set was used to generate a significant number of runs across all problems. The problem set contains instances of problems with a range of properties including low or moderate conditioning, unimodal, multimodal and both strong and weak global structure examples In this paper, we use the noiseless variant of the Black Box Optimisation Benchmarking function set (Finck et al. 2010). It is important to note that while the BBOB problems are named "black box", the optima and function values are known for each of the functions beforehand. The search space for these problems is typically box-constrained, meaning that the solutions are restricted to lie within a defined region. For BBOB functions, this is often represented as $[-5, 5]^d$, where d is the dimensionality of the problem. This is what was used in this paper. The approach is the same whether there is a unique global optimum or multiple global optima. On different runs the algorithm may converge on different optima and this will be reflected in the statistics calculated across all optima simultaneously. This allows us to identify key, influential variables over multiple runs.

To generate the trajectories for analysis, all 24 of the available BBOB functions from the BBOBtorch (Valkovič 2021) implementation were used. This provided a flexible framework that allowed for rapid implementation of the problem set. While all 24 functions were tested, Section 5 outlines the subset of interest for which further analysis was performed to illustrate our methods.

Each BBOB function represents an instance of a problem, and for our study, all algorithm runs were executed on the same instance of each problem to maintain consistency. This was achieved by using the same BBOBtorch seed parameter value of 0, ensuring that all algorithms solved the same instance with the same problem. By doing so, we generate a dataset of 100 independent runs by each algorithm on the same problem landscape, allowing for the detection of search behaviours common across all runs with randomised starting positions. These problems

were solved using a selection of four population-based metaheuristics, each solving a problem a total of 100 times to generate a reasonable number of optimisation runs for data collection. These algorithms were a Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm, a Differential Evolution algorithm (DE), a Genetic Algorithm (GA) and a Particle Swarm optimisation algorithm (PSO).

Each optimisation run was initialized with a randomly generated starting population for all runs across all problems and algorithms. This allows for the comparison of algorithm behaviour on the same problem function instance with the same optimum value but all starting points were randomized. This approach was chosen to analyse whether, given a random starting position, the algorithms would show similar search behaviours that were common to a specific problem.

2.2 Algorithms

To generate the datasets used in this paper, a selection of population-based metaheuristics were used. These were selected as they provide a significant range of search techniques while remaining approachable enough to relate detected features to their designed search behaviour. The algorithm implementations used were created and packaged as part of the PYMOO (Blank and Deb 2020) Python package.

The settings used for our optimisation runs included the number of runs, maximum generation count, population size, and problem dimensionality. We used 100 runs per algorithm-problem pair with 300 generations, ensuring ample data for analysis. A fixed population size was chosen to standardize the trajectory structures and ensure consistent data points across all algorithm and problem pairs for the decomposition process. While adaptive population size adjustments, as seen in (Poláková et al. 2019), can enhance performance, it may not be advantageous for 10-dimensional problems. We settled on a population size of 50, considering the diversity of algorithms. As shown in (Piotrowski et al. 2020), PSOs can benefit from larger-than-traditional population sizes. Thus, we opted for the higher end of their "classical" range, maintaining consistency across all algorithms with the value of 50 which provides a large enough selection pool for all algorithms used. Algorithm-specific run settings were taken from the recommended settings for each including crossover rate, particle velocity and algorithm variant. It is important to highlight that this study did not focus on optimising the algorithms' performance. Our primary concern was generating higher-quality solutions to facilitate our analysis. Consequently, whenever possible, we adhered to the default values and operators provided in the PYMOO documentation. This was done as the experiments in this paper are not intended to compare like-for-like algorithm performance or rate the PYMOO implementations but rather to use it as a framework to generate datasets showcasing the differing behaviours of the selected algorithms under their default, natural conditions.

2.2.1 ***Covariance Matrix Adaptation Evolution Strategy.*** The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen and Ostermeier 2001) is a form of evolutionary algorithm designed to be a stochastic derivative-free numerical optimisation method for non-linear or non-convex continuous optimisation problems including often difficult problems such as those with rugged fitness landscapes. Equation (1) outlines how a population of solutions may be sampled for any given generation.

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

Early termination criteria were disabled to allow for longer trajectories to be created. Examples include extending the function evaluation termination value to allow for a full 300 generations. Here, the parameters are as defined by the author:

- $\sim$ denotes the same distribution on both sides.

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

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

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

2.2.2 ***Differential Evolution.*** This method of optimisation involves the perturbation of a vector representation of populations of solutions using vector differences. This process of moving from one population of solutions to the next is outlined in Equation (2) and is based on the single-objective optimisation algorithm set in (Price et al. 2005).

$$v = x\pi_1 + F \cdot (x\pi_2 - x\pi_3) \tag{2}$$

Here v is the vector representing the "donor" vector that is created by the mutation and crossover process. $x\pi_1$, $x\pi_2$ and $x\pi_3$ are solutions in π, a collection of three randomly selected, mutually exclusive solutions in the current population. Once complete, a second crossover event takes place between a given individual solution and the donor vector v. As noted in the PYMOO implementation, this can be binomial, uniform or exponential. This process is described in detail in (Das and Suganthan 2011). In this paper we use the "DE/rand/1/bin" variant of the DE algorithm. This means that the base vectors are randomly chosen, one vector difference is used to mutate the population and binomial distribution is used. The crossover rate for the DE was set to 0.9. This value was selected as it was a recommended value from the documentation. It was noted that a lower crossover rate (CR) value tends to help with optimising separable functions however a default of 0.9 for use in a broader set of functions was used.

2.2.3 ***Genetic Algorithm.*** In this paper the implementation of a Genetic Algorithm used was a $(\mu+\lambda)$ algorithm, where μ is an initial population and λ is the population of offspring solutions once the internal operators have been used. This algorithm generates each successive population of solutions by performing the Selection, Crossover and Mutation operations on the parent population.

Here, two parent solutions were randomly selected from the population using tournament selection. Crossover was performed using a simulated binary crossover (SBX) method in which "Real values can be represented by a

binary notation and then single-point crossovers can be performed…" (Blank 2020). A polynomial function, (Deb and Deb 2014) was used to handle mutation, details of which can be found in (Deb et al. 2007). The implementation of this allowed for the use of the *eta* parameter. As this SBX implementation uses an exponential distribution to simulate binary crossover, the exponential distribution can be adjusted via this parameter. In the mutation operator, the higher the value is set, the more similar and less mutated the child solution will be. The eta parameter was set to 3.0.

2.2.4 ***Particle Swarm Optimisation.*** This involves the use of multiple particles that each represent a candidate solution and tracking their positional change through the search space. Their search is controlled through the application of a calculated velocity and direction that is dependent on the swarm's current best solution and neighbouring particles' performance. The PYMOO implementation of the PSO we used is based on (Kennedy and Eberhart 2009) with alterations to allow the values for c_1 and c_2 to be updated at the same time. Equation (3) shows the formula for how the velocity of each particle is updated between generations, as outlined in (Zhan et al. 2009).

$$V_d^{(i)} = \omega\, V_d^{(i)} \;+\; c_1\, r_1 \left(P_d^{(i)} - X_d^{(i)}\right) \;+\; c_2\, r_2 \left(G_d^{(i)} - X_d^{(i)}\right) \tag{3}$$

$$X_d^{(i)} = X_d^{(i)} + V_d^{(i)} \tag{4}$$

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

Equation (4) displays how the position of each particle is also updated between populations. Here, we set the ω value to 0.9, c_1 and c_2 were set to 2.0. The Adaptive setting was set to TRUE, specifying that ω, c_1, and c_2 are changed dynamically over time based on the spread from the global optimum. The initial velocity of each particle was set to a random value between 0 and 1 and the maximum velocity rate was set to 0.2

3 POPULATION COLLECTION

Defining a structure for the populations of solutions generated through optimisation is an important step in the pipeline. Each optimisation run generates a series of solutions, structured by generation and each generation represents the algorithm's current position in the search. In this paper, we define a trajectory T as a collection of EA solution populations X ordered by their generation g as shown in Equation (5). This notation allows us to further define an EA search trajectory as a collection of $\mathbb{R}^{gNn}$ solutions ordered by the number of generations, g, the population size N and the problem dimension n.

$$\begin{aligned} T &= [X_1, \ldots, X_g]^\intercal \\ X &= \{x^1, \ldots, x^N\}^\intercal \\ x &= [x_1, \ldots, x_n] \in \mathbb{R}^n \end{aligned} \tag{5}$$

3.1 Fitness Quartiles

Here we outline the methodology for determining the generation number at which a specific fitness threshold, denoted by f_g^*, is reached within an optimisation run. The fitness threshold achievement is used as a measure for understanding the progress of an optimisation process. The process of calculating the generation number at which this occurs can be seen in Equations (6) to (8).

$$f_g^* = \min_{x \in X_g} F(x_g) \tag{6}$$

$$\Delta f^* = f_0^* - f_l^* \tag{7}$$

$$g_q = \min\{g : f_0^* - f_g^* \geq q\Delta f^*\} \tag{8}$$

In these equations, we show how the calculation of the generation number at which a certain fitness threshold can be achieved in a given optimisation run. Here, $F(X_g)$ is the collection of fitness values of all solutions at generation g. f_g^* represents the best, in this case, minimum, fitness value in a given generation. For our calculations, we use f_0^* to represent the initial best-found solution in the starting population of an optimisation run. Δf^* is the total reduction in fitness from the initial best solution to the final solution, here shown as f_l^*. This final solution represents the best-so-far solution found by the end of the optimisation run. The fitness thresholds are represented by q. In this paper, we use the thresholds [0.25, 0.5, 0.75, 0.99] for each of the fitness quartiles. These thresholds represent an overall fitness reduction of 25%, 50%, 75% and 99%. For each of these thresholds, we calculate g_q which is the generation at which threshold q is achieved. This generation number is then averaged across all runs for each algorithm and problem pair to determine the mean generation at which each of the thresholds is achieved. This is then used to partition the optimisation runs into quartiles for later analysis, in which we can focus on specific areas of interest within the trajectories.

3.2 Dimension Reduction

There are many possible methods of dimension reduction when dealing with algorithm trajectories available. These methods can broadly be broken down into convex and non-convex methods as noted in (Van Der Maaten et al. 2009) from which Figure 2 is taken. In that paper, convex and non-convex methods are defined as "Convex techniques optimise an objective function that does not contain any local optima, whereas non-convex techniques optimise objective functions that do contain local optima". Examples of alternative techniques are Maximum Variance Unfolding (MVU) (Weinberger et al. 2004) and Stochastic Neighbourhood Embedding (SNE) (Hinton and Roweis 2002). Examples from Figure 2 include Sammon Mapping which is a non-linear dimensionality reduction method which aims to represent high-dimensional data in a lower-dimensional space while preserving the pairwise distances between points. This is done by minimizing the difference between the original distances in the high-dimensional space and the distances in the created sub-spaces. Similar to PCA, this method emphasises the preservation of any structure present in the source data into the subspace.

Locally Linear Embedding (LLE) is a manifold learning technique that seeks to preserve the local properties of data while reducing its dimensionality. Introduced in (Roweis and Saul 2000), LLE operates by computing the weights that reconstruct a data point from its neighbours in the high-dimensional space. It then uses these weights to embed the data into a lower-dimensional space, ensuring that the local geometric properties are retained. Unlike global techniques like PCA, LLE focuses on capturing the local structure.

ISOMAP is another non-linear dimensionality reduction method that extends the idea of classical multidimensional scaling (MDS) by incorporating geodesic distances on the data manifold. Proposed in (Tenenbaum et al.

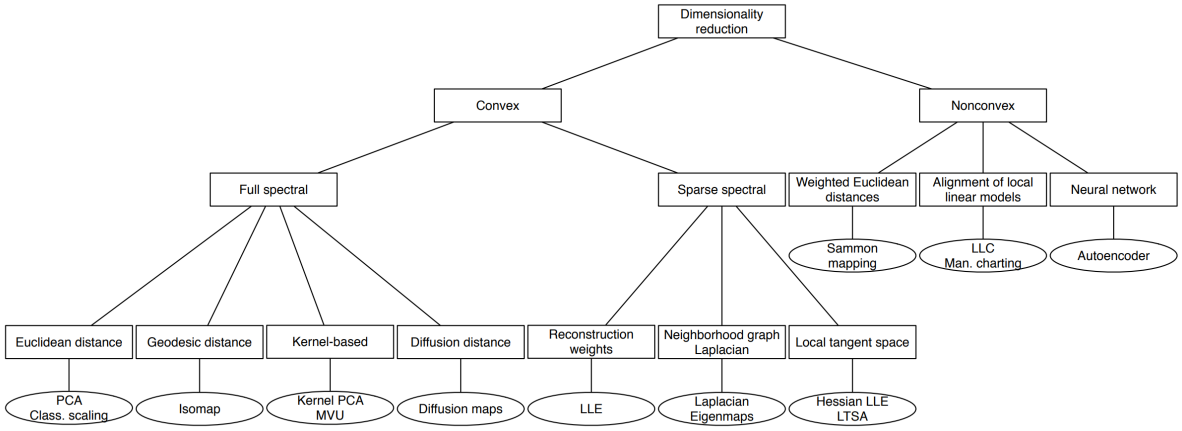


Fig. 2. Dimension Reduction Method Taxonomies - Laurens Van Der Maaten, et al, 2009. (Van Der Maaten et al. 2009

2000), ISOMAP aims to preserve the intrinsic geometry of the data by estimating the geodesic distances between all pairs of data points. It achieves this by constructing a neighbourhood graph and then computing the shortest paths between nodes.

Alternative data projection methods, such as t-distributed Stochastic Neighbourhood Embedding (t-SNE) (Van der Maaten and Hinton 2008), offer analysis capabilities similar to PCA. This popular method can similarly be used to project the trajectories into a lower-dimension subspace; however, t-SNE's stochastic nature means repeated applications on the same dataset can yield different results, posing a challenge for our experiments. Noted in a 2015 survey paper on dimension reduction techniques (Cunningham and Ghahramani 2015), t-SNE has the potential to result in distortion. Additionally, while t-SNE emphasises inter-cluster separations, it might disrupt key features from the original datasets (Husnain et al. 2021), potentially affecting the features we aim to extract.

While ISOMAP, Sammon Mapping and LLE excel in preserving non-linear structures, something PCA cannot do, our focus is on features spanning the entire trajectory with an emphasis on global over local variance. PCA was our selected method of dimension reduction for trajectory analysis because of its deterministic nature and its proficiency in capturing global variance. The preservation of geometric structure by PCA allows behaviours in its subspace to be related to those in the input data on a generation-by-generation basis, making it more applicable for our purposes than the other methods.

3.3 Principal Component Analysis

The application of PCA to our data results in the creation of a new subspace in which the axes are linear combinations of original variables. The coefficients of these linear combinations are calculated using the Singular Value Decomposition (SVD) of the scaled and mean-centred input data. This results in a set of latent variables or Principal Components (PCs) which are directional vectors calculated to maximise variance. The orthonormal matrix generated has orthogonal column vectors which act as the new axes in the subspace, defined in this paper as P.

$$P = \left[p^1, \ldots, p^m\right]^\intercal, m \leq n \tag{9}$$
$$p^i = \left[p_1^i, \ldots, p_n^i\right]$$

$$\widetilde{T} = TP \tag{10}$$
$$\widetilde{t} = \left[\widetilde{t}_1, \ldots, \widetilde{t}_m\right]$$

$$\widetilde{x} = \sum_{i=1}^{m} (x \cdot p_i) p_i = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} x_j \cdot p_j^i \right) p^i \tag{11}$$
$$\widetilde{X} = \left[\widetilde{x}^1, \ldots, \widetilde{x}^N\right]^\intercal$$
$$\widetilde{T} = \left[\widetilde{X}_1, \ldots, \widetilde{X}_g\right]^\intercal$$

This results in a number of components, m, created which is less than or equal to the number of dimensions in the original problem n. Each component consists of n coefficients which link our original n dimensional search space to an m dimensional subspace because each PC creates a hyperplane in the original coordinate space in which the data points were created, shown in Equation (9). To project the original data into the PC subspaces, the resulting "Scores" can be calculated by the multiplication of the trajectory by these new components as in Equation (10). Using this m dimension subspace of $\mathbb{R}^n$, we can define a projected trajectory as a collection of $\mathbb{R}^{gNm}$ solutions ordered by g, similarly structured to our original trajectory definition. To project the original dataset using this method, the process is shown in Equation (11). The equation describes how a single solution x can be approximated ($\widetilde{x}$) using the first m principal components. The term $x \cdot p_i$ is the dot product of the solution x with the i^{th} principal component p_i. This dot product gives the projection of x onto p_i, and multiplying it by p_i reconstructs the contribution of p_i to the approximation of x. The summation over m components provides the complete approximation, or projection, of x in the reduced space. By using m components such that $m < n$ we achieve the projection of the solution to the lower-dimensional subspace for use in later analysis. The equation then shows how this maps to the generation of solutions, $\widetilde{X}$, and the search trajectory $\widetilde{T}$.

Algorithm 1 PCA on Search Trajectory T

1: Compute the mean of each variable (x) across all populations (X) in T to get μ
2: Center the data: $\overline{X} = X - \mu$
3: Compute the covariance matrix: $S = \frac{1}{g \times N - 1} \overline{X}^T \overline{X}$
4: Perform decomposition on S to get eigenvalues λ_i and eigenvectors p_i
5: Select the top m eigenvectors to form matrix P
6: Project the data: $\widetilde{T} = TP$
7: Return $\widetilde{T}$

To illustrate this, Algorithm 1 shows how each individual optimisation run for each algorithm-problem pair is used as the input data to this process. The resulting components P are then used to transform the original data in that run to its projected version as shown in Equation (10). For all algorithm-problem pairs used in our work, the process is repeated for all 100 runs for each pair, resulting in a collection of 100 individual, projected optimisation runs $\widetilde{T}$. As an example, shown in Figure 3, all solutions generated by solving a two-dimensional version of the Sphere problem using the GA outlined earlier are used in this process. The resulting components and associated coefficients are used in our analysis methods.

4 FEATURE EXTRACTION

The techniques used in this paper aim to generate a set of features that associate changes in variable values within a trajectory to changes in fitness. These features are mined from the projected data by taking advantage of the search trajectory decomposition. With the trajectory data projected to the PCA-generated subspace, it is possible to derive features from both the input data and the scores created by this projection.

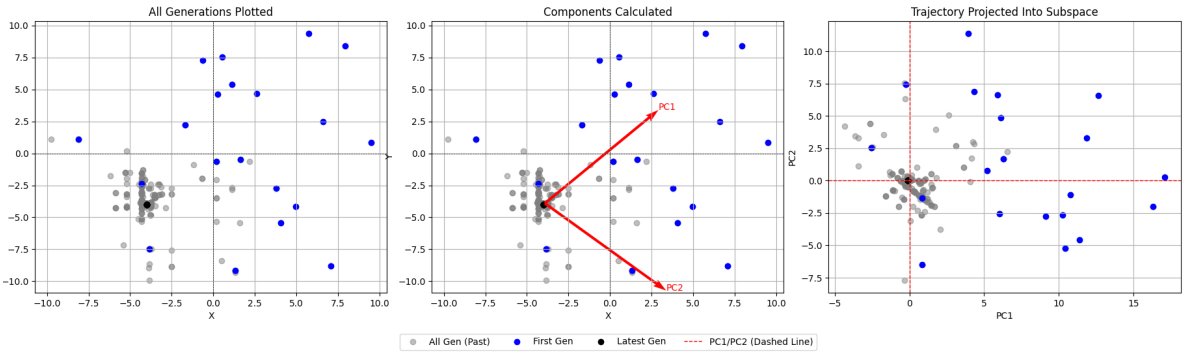


Fig. 3. Trajectory Projection Illustrative Example

4.1 Component Coefficients

The resulting directional vectors from the application of PCA to the datasets as shown in Equation (9) are a set of m, $n \times 1$ orthonormal vectors. The elements in these p^i vectors $[p_1^i, \ldots, p_n^i]$, also known as our PC coefficients, can be considered the weighting of each variable. These coefficients of a PC describe the magnitude of contribution that variable has to that principle component. The higher their absolute value, the higher their influence was in the calculation of that best-fit hyperplane such that it maximizes variance across the dataset. The signs of these coefficients indicate the direction between the PC and the variable in terms of negative and positive correlation. These features are useful as they represent each variable's contribution to the calculation of the component vector after the data has been centred and scaled. This relationship is useful as it can serve as an indication to which variables are most responsible for a solutions projection to a PC.

4.2 Mean Variable Contributions

The projection of the dataset into the PC subspace results in a set of m-dimensional vectors of $x \cdot p^i$ values, known as scores, which are the projected solutions coordinates in that space. Each of the terms $p_j^i x_j^k$ in Equation (11) represents the relative contribution variable x_j has on the overall score of solution $\widetilde{x}$ across the component p^i. A solution in which all n variable values are close to the dataset's mean will have a score value close to zero. A larger, positive score value suggests that the variables are above the mean, driving the score value further from zero. We can calculate the impact each variable has within a solution in determining the score across each component by calculating the contribution value as seen in Equation (12).

$$C_i' = \sum_{k=1}^{N} \left(\frac{\left(\sum_{j=1}^{n} p_j^i x_j^k \right)}{N} \right) p^i. \tag{12}$$

Here, C_i' is a vector of mean variable contributions across component p^i of any given generation of size N solutions with dimension n. We calculate this value for each solution in a population and take the mean value for each variable. This provides us with a method for describing the influence each variable in the problem has in determining the current population's overall position in the subspace across component p^i. By calculating this value for each variable, across a given PC at any generation, we can use this metric to indicate the relative influence of each variable at specific moments in a search trajectory. The metric is calculated by taking the mean

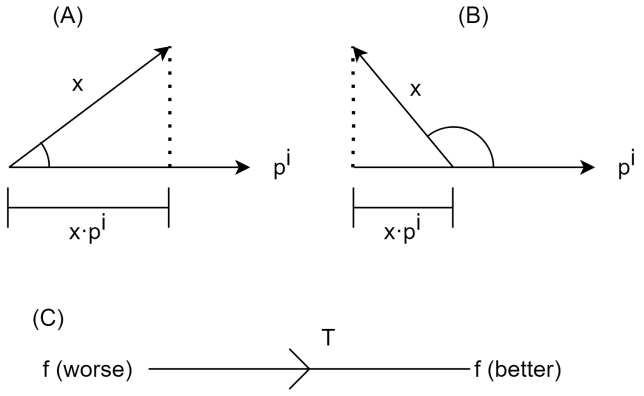


Fig. 4. Solution Vector to Component Alignment and Fitness Gradient Illustrative Example

influence of each variable across multiple runs, with emphasis on stability by considering a large number of runs with varying starting positions.

4.3 Proportional Variable Alignment to Changing Fitness

The PC coefficients that define the components in the subspace can be used to describe the overall structure of the dataset. Solutions with above-average component scores exhibit an above-average projection onto the corresponding component due to a below-average angle between the solution and that component. The direction of projection may align either in the same or opposite direction as the component itself, which could be positively or negatively oriented concerning fitness improvement. An example of this is shown in Figure 4A, 4B, and 4C. We know that the direction of travel of the algorithms Search Trajectories (T) tend to point from initially worse fitness to better fitness sets of solutions, represented in Figure 4C.

The score of a solution represents the distance from the origin to the perpendicular projection across a given PC. For example, in Figure 4A, where a solution aligns with a PC, the score would be positive. Conversely, in Figure 4B, where alignment is absent, the score would be negative.

The orientation of PCs with respect to fitness improvement depends on the results of decomposing T. If a PC points away from better fitness solutions, a negative score would still be considered aligned since the solution x remains oriented towards better fitness solutions. This method aids in identifying the alignment of specific solution variables with improved fitness solutions, based on their projections onto PCs.

We can identify if a variable in a solution aligns with better fitness based on its projection onto PCs. Shown in Equation (13), this determines the sign of each variable in solution x based on its score ($x \cdot p^i$) across PC p_i. Then, we compare each variable value x_j with its mean value across the trajectory T_j as shown in Equation (14). As the components were calculated from the scaled and mean-centred data, a positive x_j will be above the mean and a negative will be below. This returns a binary vector $G_i(x_j)$ with 1 if this is true and 0 otherwise.

$$V_j^i(x) = \begin{Bmatrix} 1 \text{ if } & x \cdot p^i p_j^i \geq 0 \\ -1 & \text{Otherwise} \end{Bmatrix} \tag{13}$$

$$G_i(x_j) = \begin{Bmatrix} 1 \text{ if } & \left(V_j^i(x) \geq 0 \text{ and } x_j \geq \bar{T}_j\right) \\ 1 \text{ if } & \left(V_j^i(x) \leq 0 \text{ and } x_j \leq \bar{T}_j\right) \\ 0 & \text{Otherwise} \end{Bmatrix} \tag{14}$$

This process is repeated across all components for all variables in a solution and the mean value per component is taken. Equation (15) shows how we calculate a total value for each solution in terms of the number of variables aligned to any given component. This is to provide a single value representing the proportion of variables aligned in a generation of solutions.

$$PG_i = \frac{\sum_{k=1}^{N} G_i(x^k)}{N} \tag{15}$$

Where PG_i is a vector representing the proportion of times that G_i in P_i for a generation of solutions was true across all n dimensions. This calculation is performed on all generations within a trajectory to measure how this proportion changes over time. The higher the mean alignment value, the better the indication as to how well the raw data is fitted by the component. We use this value to determine if one component is fitting the data better than others and measure the proportion of variables that are aligned with higher-quality solutions. We can use this to select the best-fitting component at any given generation and focus on how the variables in a population of solutions contribute to the position across that component. This can be applied to both the entire solution or a subset of variables of high interest. In Equation (16) we show how we can use the subset b in the same method.

$$PGS_i = \frac{\sum_{k=1}^{N} \sum_{j=1}^{B} G_i(x^k_{b_j})}{N} \tag{16}$$

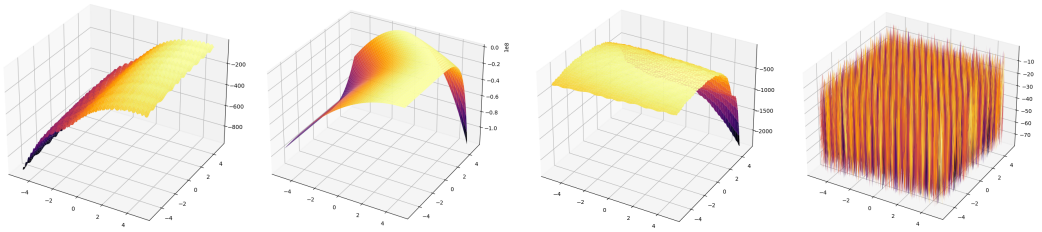
Let $\{b_1, \ldots b_m\}$ be a subset of $\{1, \ldots, n\}$ of size $m \leq n$. This is used to calculate PGS_i, a vector representing the proportion of times that G_i in P_i for a generation of solutions was true when solutions are projected onto the m-dimensional subspace spanned by the components indexed by $\{b_1, \ldots b_m\}$. For calculating PGS_i, $\{b_1, \ldots b_m\}$ will consist of that subset of variables with the m largest absolute coefficient values for component i. This is because they represent the most influential variables in determining a solutions score in that component and allows us to compare these results to the values calculated from using all n dimensions.

5 EXPLANATION GENERATION

The trajectory datasets used for this paper were assembled from 9,600 runs across 24 BBOB problem instances, generated by the four metaheuristics selected. Each problem instance used was initialized with an identical set of parameters. This meant that every instance of a specific BBOB problem had the same optima and fitness landscape across all 100 runs and four algorithms, allowing the trajectories to be compared directly for each algorithm.

5.1 Optimisation Problem Subset

To illustrate the results of our analysis, a subset of four BBOB problems was selected. This selection was comprised of one function from each main group of problem types outlined in (Finck et al. 2010). Each problem type is broadly defined by their features as detailed in (Finck et al. 2010). The first of the four selected was the Rastrigin Function (F3). This provided an example of a separable function from the collection. The second selected was the Bent Cigar Function (F12) to provide an example of a function with high conditioning while remaining uni-modal. The third selected is a modification of the Rastrigin Function (F15) as an example of a multi-model function with an "adequate" global structure as described in the functions implementation documentation. This modification alters the regularity and symmetry of the original function. Lastly, the Katsuura Function (F23) was selected to provide trajectories for a multi-model function with "weak" global structure as again described in the functions implementation documentation. This last function was also selected as an example of a considerably more rugged landscape with a large number of global optima. This selection provided a representative sample of the functions available in the BBOB collection. While these four have been selected to illustrate our methodologies, the results



(a) Function F3 Rastrigin (b) Function F12 Bent Cigar (c) Function F15 Rastrigin (Multi-Modal) (d) Function F23 Katsuura

Fig. 5. Three-D Landscapes of Selected BBOB Problems for Illustration

of the following analysis, including all trajectories, on all 24 problems and four algorithms are available online (Fyvie et al.). An extract of these extended results showing the top four contributing variables can be found in Appendix A.

Shown in Figure 5 are plots illustrating the landscape of each of the representative functions in three dimensions.

5.2 Algorithm Performance

The results of the optimisation runs are summarized in Table 1. Shown in this table are the algorithms used and the problems solved. The table also shows F-Opt, which is the optimal value for each problem instance used. The median best fitness across 100 runs is calculated and presented. This value is calculated by taking the best-found solution in each of the 100 runs and calculating the median fitness. The Min and Max values are the minimum and maximum values found in that same collection of the best 100 solutions. We also show the standard deviation of that collection and the difference in objective value. This is a measure of the distance, in terms of fitness value in the objective space, between the median best fitness and the problem's optimal value. Highlighted in bold is the best-performing algorithm for each problem, measured by the distance to the optima. The final population fitness distributions, after the removal of outliers, is visualised in Figure 6.

These results show that, for the F3 problem, the GA stands out. It achieved the lowest mean fitness distance to the optima with a value of 0.01. This indicates that it consistently approaches the optima fitness value of the F3 function. In relation to the other algorithms, the GA also has the lowest standard deviation. For the remaining problems, the CMA-ES is highlighted as achieving the closest median fitness values to the optimal solutions. In problem F12, DE shows a slightly lower standard deviation showing that, while it did not consistently find the optimal as the CMA-ES did, it has a lower spread of fitness values. Overall, while the CMA-ES consistently achieved solutions with minimal deviation from the optimal values in three of the four problems, the algorithm run results (Table 1) demonstrate that all algorithms are capable of discovering high-quality solutions over the course of their optimisation runs.

5.3 Component Explained Variance

The components generated from the decomposition of the trajectories are ordered by the magnitude of their eigenvalues. By dividing the eigenvalue of each component by the sum of the eigenvalues we can show the percentage of variance each PC can explain in the data. Shown in Figure 7 are the explained variance ratio results across 10 components for each pair of algorithms and problems. This figure shows the diminishing level of variance explained as further components are added.

5.4 Fitness Quartile Windows

To allow for a fair comparison of variable contribution changes over the course of a search trajectory, we calculated the mean generation number in which the algorithm had achieved a set of fitness goals. This value was then rounded to the nearest whole number for the purposes of sub-setting our data by generation. The goals were set at 25%, 50%, 75% and 99% of total

Alg	Prob	F-Opt	Median Best F.	Min	Max	Std	Dist. To. Opt
CMAES	F3	129.88	137.79	130.87	152.62	3.48	7.91
DE	F3	129.88	153.43	145.21	162.99	4.14	23.55
GA	**F3**	**129.88**	**129.89**	**129.88**	**129.95**	**0.01**	**0.01**
PSO	F3	129.88	132.97	129.88	145.64	2.90	3.09
CMAES	**F12**	**1000.00**	**1000.00**	**1000.00**	**1006.39**	**1.28**	**0.00**
DE	F12	1000.00	1000.51	1000.09	1005.50	1.13	0.51
GA	F12	1000.00	1016.72	1001.44	1261.90	28.05	16.72
PSO	F12	1000.00	1001.15	1000.00	1010.33	2.53	1.15
CMAES	**F15**	**1000.00**	**1006.92**	**1000.00**	**1024.93**	**4.74**	**6.92**
DE	F15	1000.00	1035.73	1022.69	1046.39	5.22	35.73
GA	F15	1000.00	1025.02	1006.02	1055.37	10.87	25.02
PSO	F15	1000.00	1020.76	1004.94	1064.26	10.87	20.76
CMAES	**F23**	**129.88**	**129.92**	**129.88**	**131.62**	**0.19**	**0.04**
DE	F23	129.88	131.42	130.84	132.07	0.26	1.54
GA	F23	129.88	130.86	130.23	131.90	0.37	0.98
PSO	F23	129.88	131.40	130.48	132.16	0.36	1.52

Table 1. Algorithm Run Results

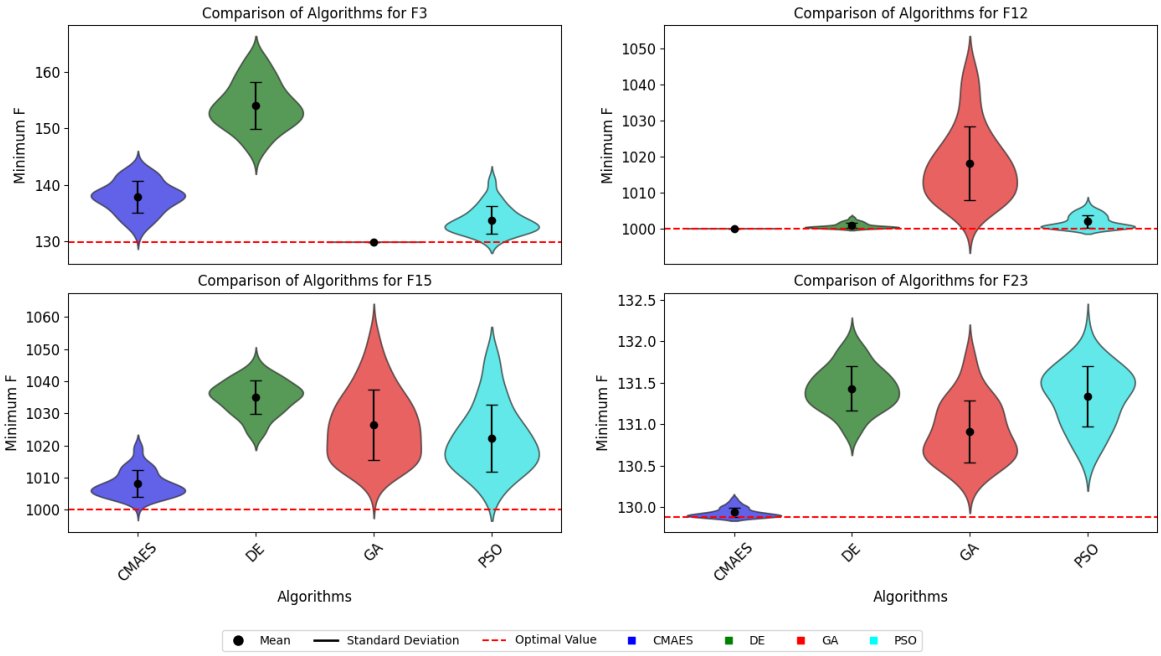


Fig. 6. Final Population Fitness Distribution by Algorithm and Problem Over 100 Runs

fitness reduction in the set of minimization problems. This was calculated for each algorithm-problem pair and presented in Table 2. These values are the mean value across all 100 runs of each pair and are calculated using Equations (6) to (8). The table

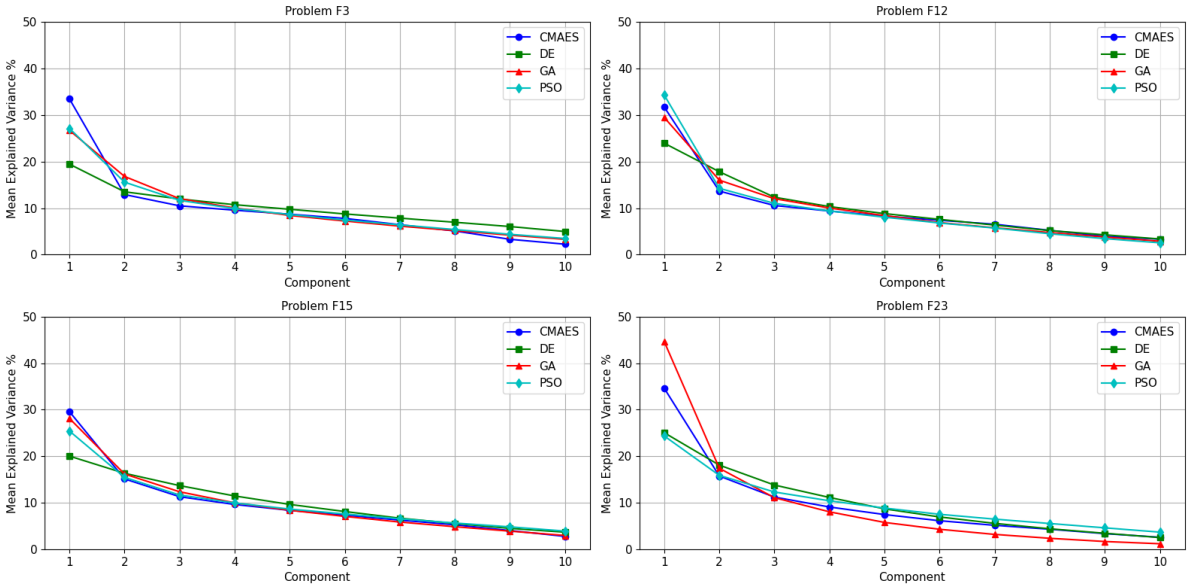


Fig. 7. Algorithm Explained Variance Ratio

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

Table 2. Generations Until Window of Fitness Achieved.

shows the mean generation that defines the fitness window. These values were used to subset the trajectories into windows based on generation number.

As previously mentioned, the CMA-ES trajectories were extended to the same length as the other three algorithms through the repetition of the final population. This allowed for the comparison and plotting of the search paths on a generation-by-generation basis across the entire search trajectory. This method was used to generate the plots showing the results of the variable alignment proportions shown in Section 5.6.

The mean variable contribution calculations may be sensitive to this extension as it would affect the mean value calculated in the final window if that target was set to 100% of fitness. By using the 99% target, the final window can be considered close to the point of convergence with less chance of being affected by a large number of low-diversity solutions at the end of the trajectory.

5.5 Mean Variable Contributions

Using the fitness windows from each algorithm-problem pair, the mean variable contribution (C_i') was calculated across the first three PCs. To achieve this, the generation bounds identified for each fitness quartile were utilised to partition the individual optimisation runs into four distinct sections. Then, contribution values for each variable were computed within each window, and the means across all 100 runs were determined for each quartile. The Mean Variable Contribution results are plotted in Figures 8 - 11.

These are the mean contribution values for each variable across the first three PCs' windows. These values illustrate a variable's relative influence on the overall position within a PC, compared to other variables. By averaging across 100 runs and segmenting fitness windows, we can highlight variables' significance in determining algorithm positions during critical trajectory phases. Contribution values aid in identifying key variables driving positional shifts towards improved fitness solutions as algorithms evolve their populations.

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

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

5.5.3 Function F15. The contribution charts seen in Figure 10 for function F15 highlight that variables [1, 2, 3, 6] have the highest contribution values across either PC1 or PC2 during the 25% window. The CME-ES, GA and PSO results show these to have the highest contribution across PC1 however the DE results show variable [2] to be considerably more contributory to PC2 than PC1. This remains true in all four quartiles for DE. While these variables are highlighted as being important for all four algorithms, the level of contribution shows variation between them. The CMA-ES, in the 75% and 99% quartiles, show how variables [4,9] grow in their level of contribution in PC2, becoming the most influential during this stage of the search. The DE results show that variables [1, 2, 3, 4, 9] become the most contributory across all three PCs by the 99% quartile. The results for DE, GA, and PSO indicate that while initially, a small subset of variables across PC1 has the highest value, by the 99% quartile, the contribution to all PCs becomes more evenly distributed.

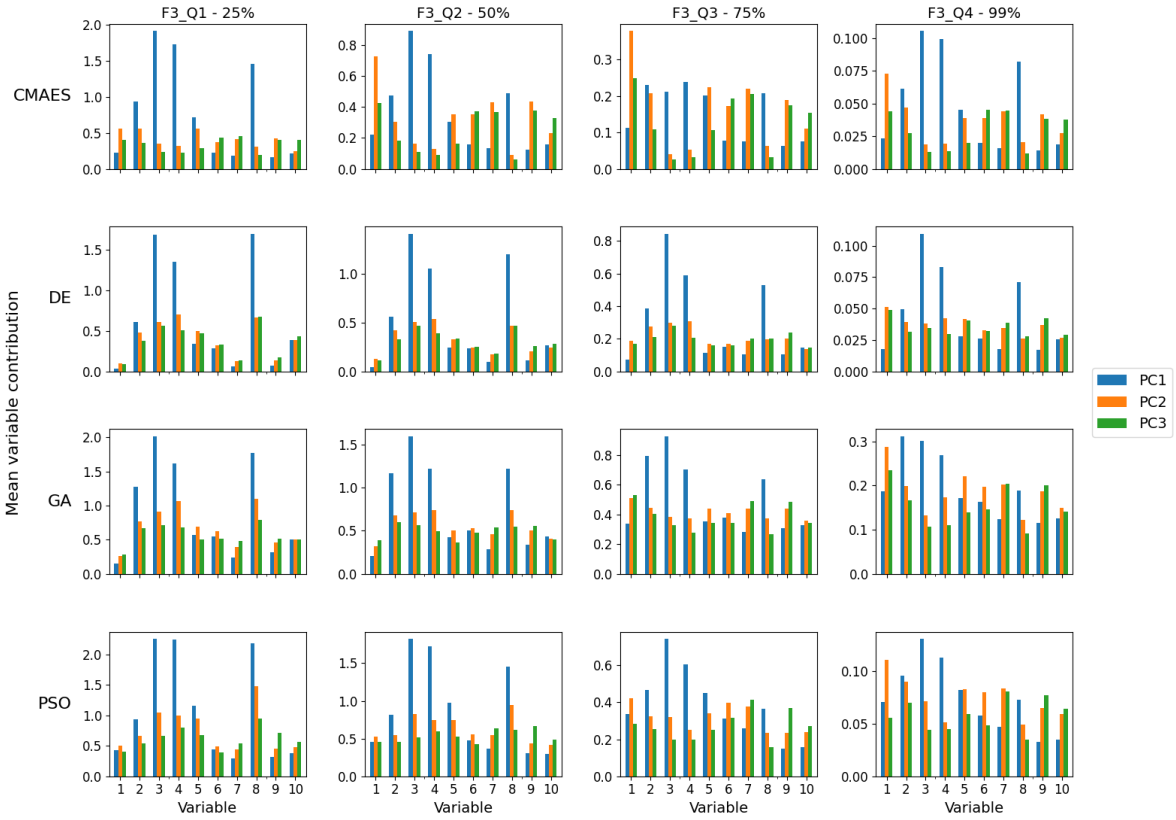


Fig. 8. F3 - Variable Contributions Across Algorithms

5.5.4 Function F23. Function F23 was selected as an example with a considerably more rugged fitness landscape. The results as shown in Figure 11 show the variable contribution calculations in the four fitness windows identified. The results for the CMA-ES, DE and PSO show lower contribution scores across all PCs when compared to the GA, with the DE results showing the lowest values of any in the 25% and 50% quartiles. The problem also has some of the highest mean generation numbers before 99% of fitness reduction was achieved across all algorithms as seen in Table 2. The results show that, in all quartiles for the CMA-ES, GA and PSO, the majority of variables are identified as having a higher contribution in PC1. The more evenly distributed values across all three PCs show that none of the algorithms determined that a specific subset of the variables contributed significantly more than others. Given the long time to achieve convergence across all algorithms, it may be concluded that no specific subset of variables was driving the search path more than others during each window.

5.6 Proportion of Aligned Variables

The Proportion of Aligned Variables (PG_i) is a measurement of the proportion of variables in the input data that are being correctly described by a PC which is a best-fit hyperplane calculated to represent maximal variance in the input data. To calculate this we used Equations (13), (14) and (15). The input data for this analysis was the extended trajectories so that all trajectories used had the same length. This was done for the purpose of visualising the results so that all plots may have the

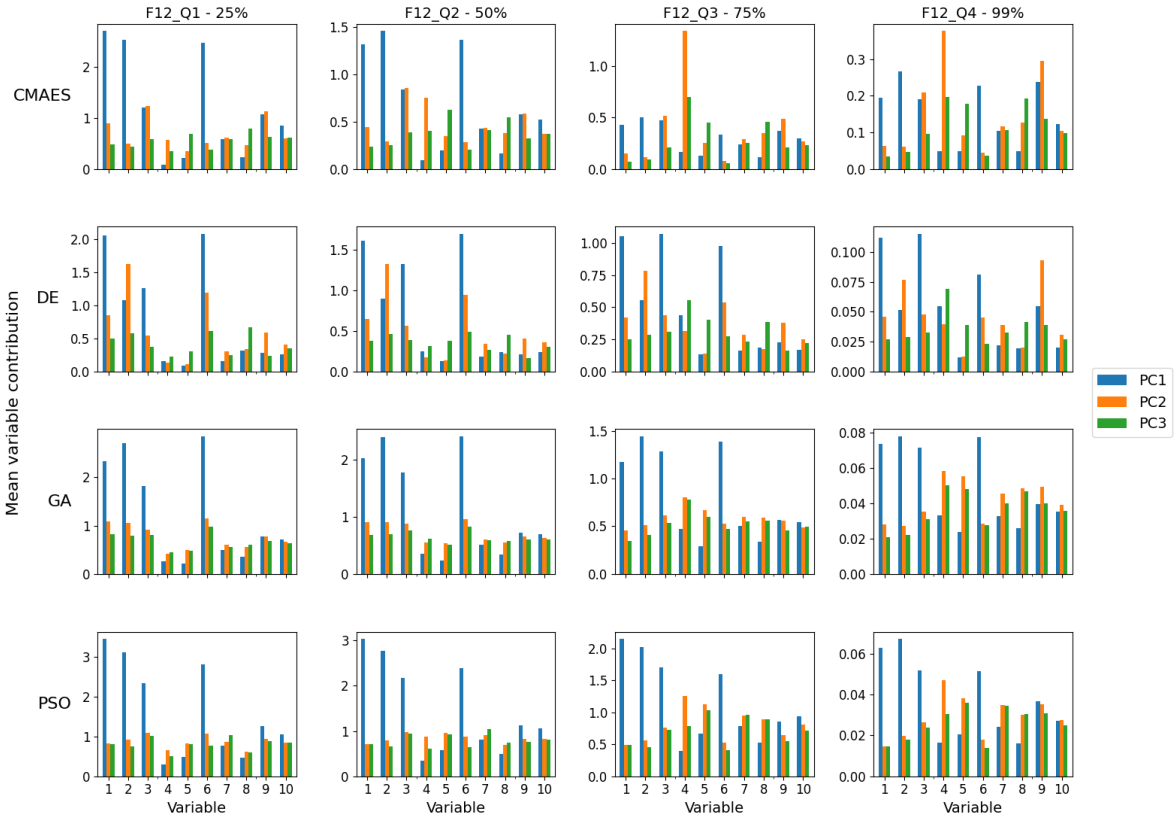


Fig. 9. F12 - Variable Contributions Across Algorithms

full range of 300 generations. Alignment is calculated from a solution vector by testing whether the variables are distributed as described by the coefficients of the first three PCs. The more variables that are shown to be in the same direction as the coefficients, the more aligned that solution vector is. This value is calculated for each variable by calculating the proportion of times that it is identified as being aligned in a generation and taking the mean value across all 100 runs.

5.6.1 Function F3. The results of calculating the alignment values for Function F3 are shown in Figure 12. Here, we see that the proportion of variables aligned to PC1 for all algorithms begins with a value between 0.65 and 0.85. Shown in red is the natural log of the mean minimum fitness achieved during the optimisation runs for each algorithm. A similar pattern of alignment values changes can be seen in all the results where PC1 begins with the highest value over the first few generations. This quickly falls toward the same values shown in the remaining PCs. The CMA-ES, GA and PSO all show moments in which the variables in PC2 have a higher alignment proportion than PC1 between generations 20 to 30. During this period, more than 50% of fitness reduction has already occurred. The DE results show that while PC1 had an initially higher proportion of variables aligned to it, this value was only 0.68, 0.08 higher than PC2. This value also fell rapidly to match those of PC2 and PC3. The Variable Contribution results show that DE did have a higher contribution score for variables 2 and 6 in PC2 with values close to those highly contributing to PC1 which may explain this behaviour.

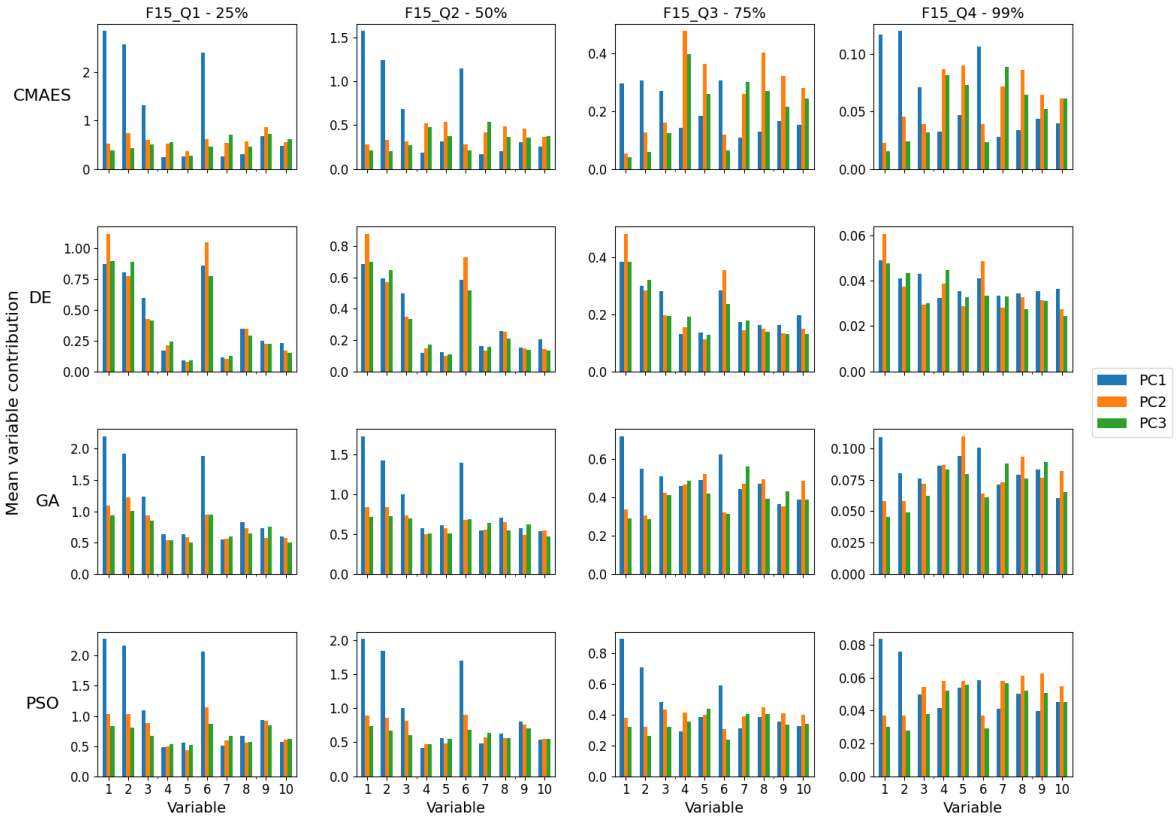


Fig. 10. F15 - Variable Contributions Across Algorithms

5.6.2 Function F12. Figure 13 shows the results for Function F12 when calculated across all generations. Here, we see a similar behaviour in both the GA and PSO with an initially high proportion in PC1 that quickly falls before becoming the highest value for the remainder of the trajectory. The DE results show that PC1 had only a slightly higher value than both PC2 and PC3 over the first few generations, all having values between 0.6 and 0.65. When comparing this to the Variable Contribution values, the DE results were the only results to have placed a high contribution to variables across all three PCs in the 25% fitness window in the same manner as was found in the F3 results. The CMA-ES results show an initially high alignment in PC1 with a value of 0.88. This rapidly fell to 0.60 over the course of three generations, after which PC2 become the most highly aligned component in terms of variable values. From generations 100 onward PC1 and PC2 alignment proportion values vary however as seen in Table 2, the CMA-ES had achieved 50% fitness reduction by generation 4 and 75% by generation 9. This occurs during and shortly after the rapid reduction in PC1 alignment value. By generation 26, 99% fitness reduction had occurred and it is possible that the erratic behaviour seen after this is an artifact of the extension of these trajectories.

5.6.3 Function F15. Function F15 results are shown in Figure 14. The GA and PSO results show similar behaviour as PC1 begins with a higher than than the other PCs before dropping as fitness is reduced. PC then returns to the highest value for the remainder of the trajectories. The CMA-ES results show a portion of the trajectory, between generations 10 and 130

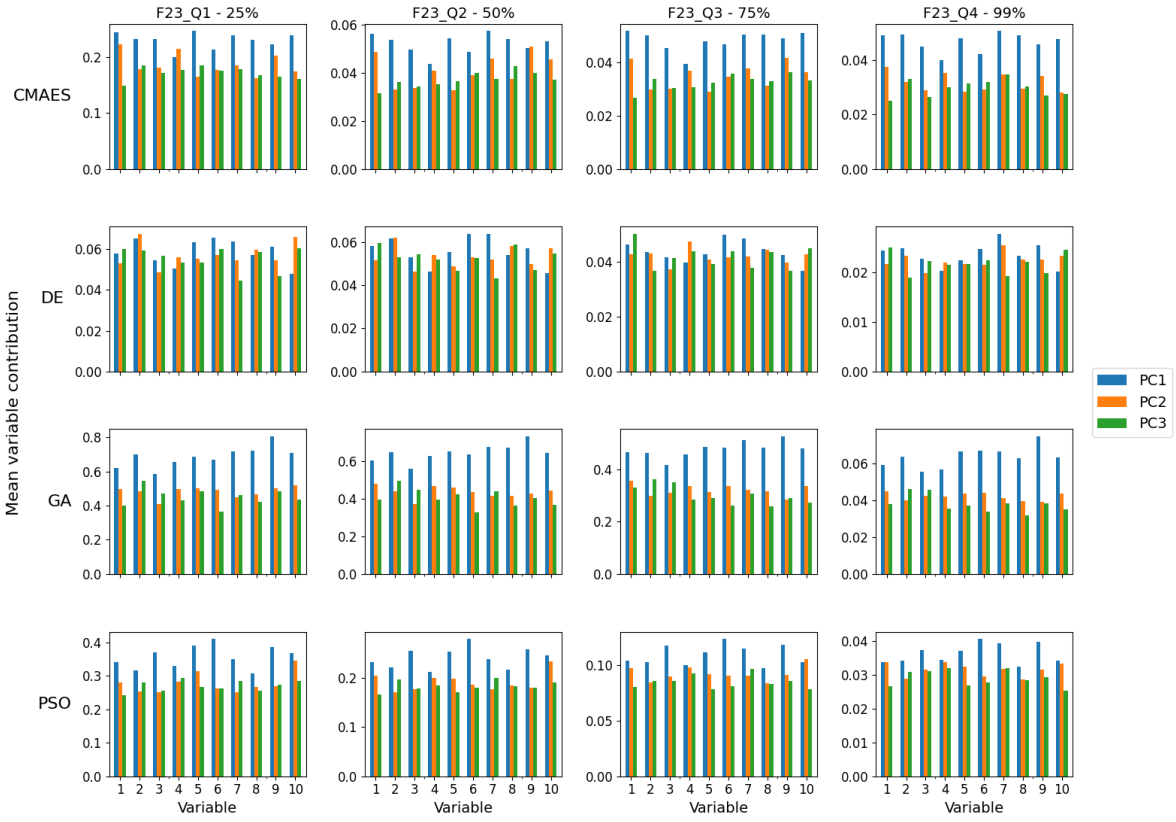


Fig. 11. F23 - Variable Contributions Across Algorithms

where PC2 has a significantly higher value than PC1 or PC2. This matches the Variable Contribution results in which the 75% and 99% results show a growing contribution from variables in PC2 and PC3 that exceed those of PC1 during the 75% window and continue to do so in the 99% window. The DE results show that overall, PC1 has the highest alignment proportion than the other PCs however as with F3, the range of values is smaller than the other algorithms with PC1 reaching a high of 0.64 and PC3 reaching a low of 0.59. The Variable Contribution results show that the search paths taken by the DE had a higher contribution from a subset of variables that slightly differed from the other three algorithms in PC1. With the smaller proportion value range and higher contribution from PC2 and PC3, this may explain why the DE required a mean of 39 generations to reduce fitness by 99%, greater than the others. Table 5 indicates that the DE did have a higher contribution in PC1 from the same subset of variables as identified by the other algorithms [1, 6, 2, 3]. The ordering however was different, with the other algorithms ordering these as [1, 2, 6, 3] with [2] being the second most contributing as opposed to [6] for the DE. This same subset was the highest contributing across all three PCs for the GA, DE and PSO. As seen in the low alignment scores and PC2-led contribution values, a longer overall run before convergence and a poorer set of results may have been the result of the DE being unable to exploit the same variable subsets as the other algorithms.

5.6.4 Function F23. The results of calculating the alignment values for Function F23 are shown in Figure 15. All algorithms show the same behaviour in terms of PC1 starting and remaining the component with the highest level of variable alignment

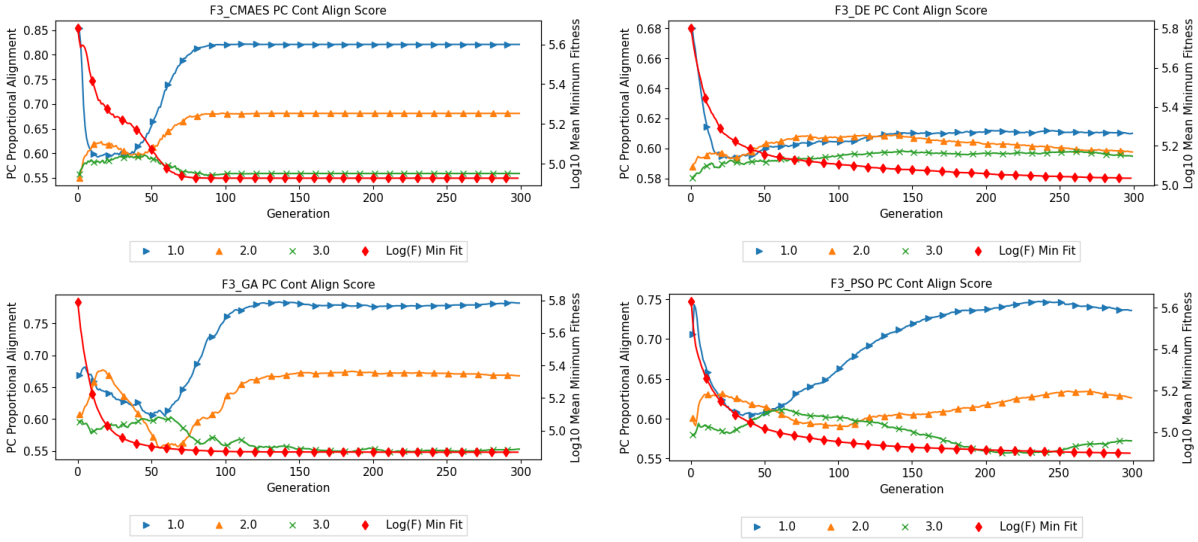


Fig. 12. F3 Alignment Proportion Results

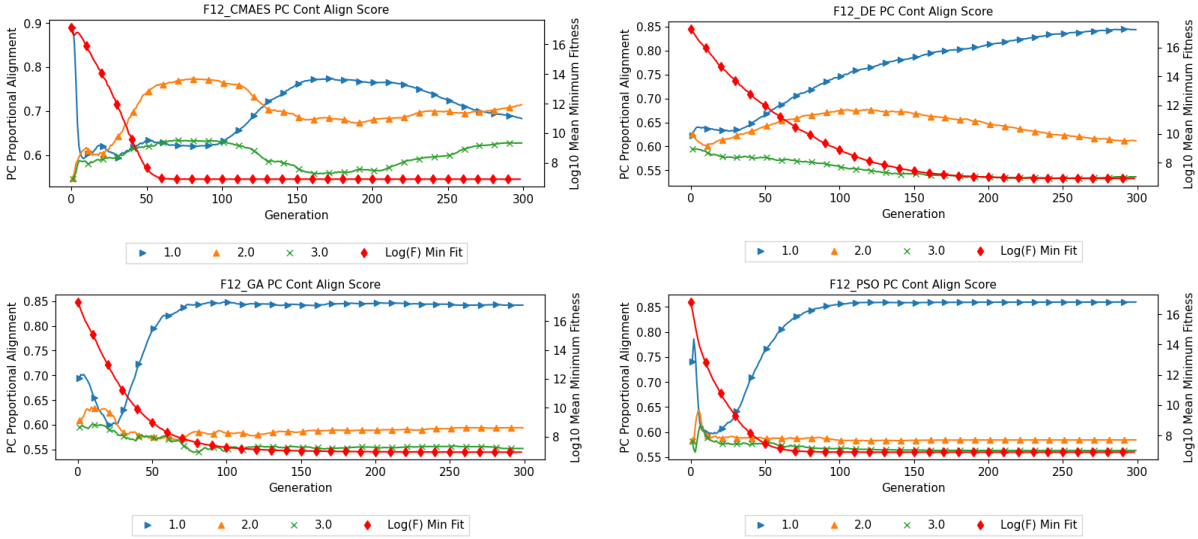


Fig. 13. F12 Alignment Proportion Results

however, the exact value varies between algorithms. The results for the CMA-ES and GA both show a high starting value in PC1. The GA value increases to near 1.0. The CMA-ES value begins at 0.75 before dropping to 0.66 and then increasing to 0.9. For the CMA-ES, DE and PSO the PC2 and PC3 values show little change however the GA values in these PCs decreased overall from 0.8 to 0.6. The Variable Contribution results for all four algorithms show the same pattern where we see a more even spread of contribution values across all variables.

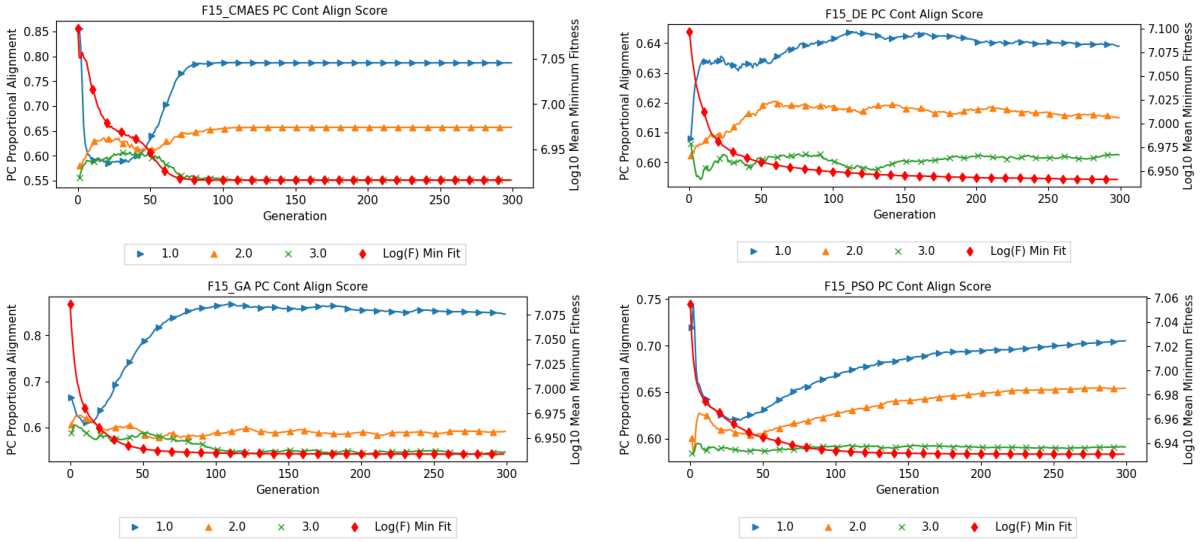


Fig. 14. F15 Alignment Proportion Results

Table 2 shows that while there are some common variables across PCs and algorithms, the same level of agreement on both variable and ranking does not appear as with other problems. One feature of the search that does stand out is the log of Mean Fitness values for the CMA-ES results. Unlike the other algorithms, this value appears to fluctuate slightly but remains high for the first 50 generations before reducing. Table 1 shows that the CMA-ES was the best-performing overall with a mean fitness difference of only 0.04 across 100 runs. The fitness windows shown in Table 2 show that the CMA-ES had taken slightly less time to achieve a 25% fitness reduction, with a mean generation of 13 however the PSO also achieved this goal within 11 generations. These results may show that the higher mean difference from the optimal solution found in the other three algorithms may have been due to them being unable to exploit the same relationships discovered by the CMA-ES earlier in the search.

The exploration of algorithm search trajectories has revealed the potential to identify subsets of variables that significantly contribute to the trajectory. By analysing these trajectories, we can determine how each variable contributes to the score value of a collection of solutions across a given Principal Component (PC), as calculated using Equation (12). When segmenting each trajectory into four windows, each representing an approximate 25% reduction in fitness, these contributions were calculated for each window. Notably, PC1 consistently demonstrated a higher contribution from solution variables compared to other PCs. This analysis facilitates the generation of explanations in terms of variable subsets that have significantly influenced the search algorithms' paths during optimisation.

The combined analysis of Variable Contribution subsets and the Proportion of Aligned Variables to Fitness Change has yielded insightful results. A notable observation is the relationship between high alignment values across a PC and significant contributions from specific variable subsets within that PC. In real-world applications, this can be used to relate what the algorithm has discovered about the variable interactions to end users; however, more specifically, in the BBOB problem set used in this paper, these are used to provide insight into the paths taken by the algorithms used.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we presented two methods developed to aid in explaining the behaviours of black-box optimisation techniques. The methods introduced were Variable Contribution and Proportional Variable Alignment, designed to mine explanatory features from algorithm search trajectories. Our comprehensive set of algorithm trajectories, derived from 24 benchmarking problems using a diverse set of metaheuristics, were projected into a lower-dimensional space using PCA. While our study

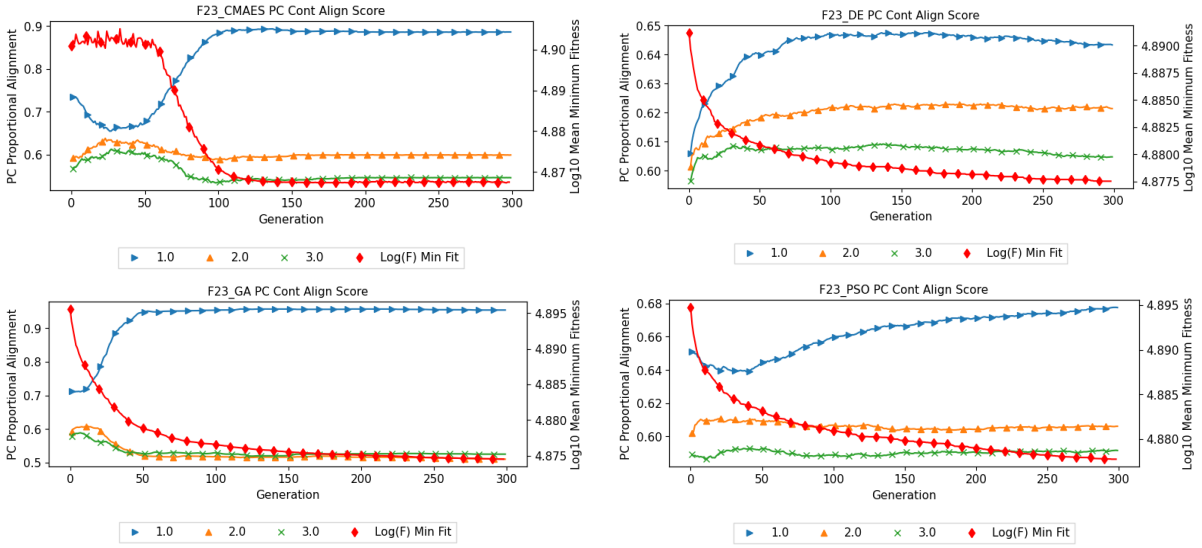


Fig. 15. F23 Alignment Proportion Results

highlights the effectiveness of PCA in capturing the global variance of search trajectories, it is essential to acknowledge the capabilities of other dimensionality reduction techniques like ISOMAP, t-SNE and LLE. These methods excel in preserving non-linear structures, a feat PCA cannot achieve. However, our emphasis on a holistic trajectory analysis, capturing features spanning the entire trajectory with a focus on global variance, made PCA the most suitable choice for our research.

The resulting metrics allow us to create an explanatory model in terms of the contribution these variable subsets have to a given PC. This linear model represents the relationship between variables as discovered by the algorithm over the course of the optimisation runs. The model allows us to compare the discoveries of different algorithms and helps highlight the variables driving the search for higher-quality solutions at different stages in a search trajectory. These features can help to explain certain aspects of algorithm search behaviour, emphasising key differences, as evident in the results for CMA-ES and DE. The results also indicate that the subsets identified during the initial 25% fitness window often remain the most influential contributors in subsequent windows. Tables 3 - 6 in Appendix A provide a comprehensive overview of the top four contributing variables for all 24 BBOB problems during the initial 25% fitness reduction. This not only offers insights into the inherent behaviours of these algorithms but also helps in making these algorithms more accessible and interpretable. This approach can identify variables with a disproportionately high impact on overall solution quality, highlighting the need for further analysis to understand their contributions. Additionally, it enables differentiation between the search behaviours of various algorithms when applied to the same set of problems. By comparing these behaviours, new EA iterations can be designed to exploit the same influential variables as other successful algorithms. Additionally, this analysis can reveal variables that are underutilised by certain algorithms, potentially offering insights into alternative search strategies and their effects on overall performance.

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

ACKNOWLEDGMENTS

This paper was written as part of a funded PhD project supported by The National Subsea Centre, Robert Gordon University, The Data Lab and BT Group plc.

REFERENCES

Jason Adair, Gabriela Ochoa, and Katherine M. Malan. Local Optima Networks for Continuous Fitness Landscapes. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 1407–1414, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367486. doi: 10.1145/3319619.3326852. URL https://doi.org/10.1145/3319619.3326852.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI. *Information Fusion*, 58:82–115, 2020. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2019.12.012. URL https://www.sciencedirect.com/science/article/pii/S1566253519308103.

J. Blank and K. Deb. Pymoo: Multi-Objective Optimization in Python. *IEEE Access*, 8:89497–89509, 2020.

Julian Blank. Crossover. https://pymoo.org/operators/crossover.html, 2020. Accessed: 2024-10-01.

Gjorgjina Cenikj, Gašper Petelin, Carola Doerr, Peter Korošec, and Tome Eftimov. DynamoRep: Trajectory-Based Population Dynamics for Classification of Black-box Optimization Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 813–821. ACM, July 2023. doi: 10.1145/3583131.3590401. URL http://dx.doi.org/10.1145/3583131.3590401.

Francisco Chicano, Gabriela Ochoa, Bilel Derbel, and Lorenzo Canonne. Local Optima Markov Chain: A New Tool for Landscape-Aware Analysis of Algorithm Dynamics. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 284–292, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701191. doi: 10.1145/3583131.3590422. URL https://doi.org/10.1145/3583131.3590422.

Raphaël Cosson, Bilel Derbel, Arnaud Liefooghe, Hernán Aguirre, Kiyoshi Tanaka, and Qingfu Zhang. Decomposition-Based Multi-objective Landscape Features and Automated Algorithm Selection. In Christine Zarges and Sébastien Verel, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 34–50, Cham, 2021. Springer International Publishing. ISBN 978-3-030-72904-2.

John P Cunningham and Zoubin Ghahramani. Linear Dimensionality Reduction: Survey, Insights, and Generalizations. *The Journal of Machine Learning Research*, 16(1):2859–2900, 2015.

Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential Evolution: A Survey of the State-of-the-Art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, 2011. doi: 10.1109/TEVC.2010.2059031.

Kalyanmoy Deb and Debayan Deb. Analysing Mutation Schemes for Real-Parameter Genetic Algorithms. *International Journal of Artificial Intelligence and Soft Computing*, 4:1–28, 2014. doi: 10.1504/IJAISC.2014.059280.

Kalyanmoy Deb, Karthik Sindhya, and Tatsuya Okabe. Self-Adaptive Simulated Binary Crossover for Real-Parameter Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '07, page 1187–1194, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936974. doi: 10.1145/1276958.1277190.

A. Duygu Arbatli and H. Levent Akin. Rule Extraction from Trained Neural Networks Using Genetic Algorithms. *Nonlinear Analysis: Theory, Methods & Applications*, 30(3):1639–1648, 1997. ISSN 0362-546X. doi: https://doi.org/10.1016/S0362-546X(96)00267-2. URL https://www.sciencedirect.com/science/article/pii/S0362546X96002672.

Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, and Rajiv Ranjan. Explainable AI (XAI): Core Ideas, Techniques, and Solutions. *ACM Comput. Surv.*, 55(9), 2023. ISSN 0360-0300. doi: 10.1145/3561048. URL 10.1145/3561048.

Alberto Fernández, Francisco Herrera, Oscar Cordón, María José del Jesus, and Francesco Marcelloni. Evolutionary Fuzzy Systems for Explainable Artificial Intelligence: Why, When, What for, and Where to? *IEEE Computational Intelligence Magazine*, 14(1):69–81, 2019.

Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. Real-Parameter Black-Box Optimization Benchmarking 2010: Presentation of the Noiseless Functions. Technical report, Technical Report 2009/20, Research Center PPE, 2010.

Martin Fyvie, John A. W. McCall, and Lee A. Christie. Telo2023_xai_data github repository. https://github.com/paperpublicationdata/TELO2023_XAI_Data.git.

Martin Fyvie, John AW McCall, and Lee A Christie. Towards Explainable Metaheuristics: PCA for Trajectory Mining in Evolutionary Algorithms. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 89–102. Springer, 2021.

Martin Fyvie, John McCall, Lee Christie, and Alexander Brownlee. Explaining a Staff Rostering Genetic Algorithm Using Sensitivity Analysis and Trajectory Analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '23 Companion, pages 1648–1656, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701207. doi: {10.1145/3583133.3596353}. URL https://doi.org/10.1145/3583133.3596353.

Petr Gajdo, Pavel Kromer, and Ivan Zelinka. Network Visualization of Population Dynamics in the Differential Evolution. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1522–1528. IEEE, 2015.

Nikolaus Hansen and Andreas Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary computation*, 9 (2):159–195, 2001.

Geoffrey E Hinton and Sam Roweis. Stochastic Neighbor Embedding. *Advances in neural information processing systems*, 15, 2002.

Martin Holena. Anomaly Explanation with Random Forests. *Journal of Machine Learning Research*, 21(110):1–25, 2020. URL http://jmlr.org/papers/v21/19-773.html.

Mujtaba Husnain, Malik Muhammad Saad Missen, Shahzad Mumtaz, Dost Muhammad Khan, Mickäel Coustaty, Muhammad Muzzamil Luqman, Jean-Marc Ogier, Hizbullah Khattak, Sikandar Ali, Ali Samad, and Shahzad Sarfraz. Urdu Handwritten Characters Data Visualization and Recognition Using Distributed Stochastic Neighborhood Embedding and Deep Network. *Complexity.*, 2021, Jan 2021. ISSN 1076-2787. doi: 10.1155/2021/4383037. URL https://doi.org/10.1155/2021/4383037.

Anja Jankovic. *Towards Online Landscape-Aware Algorithm Selection in Numerical Black-Box Optimization.* PhD thesis, Sorbonne Université, 2021.

Anja Jankovic, Diederick Vermetten, Ana Kostovska, Jacob de Nobel, Tome Eftimov, and Carola Doerr. Trajectory-Based Algorithm Selection with Warm-Starting. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2022. doi: 10.1109/CEC55065.2022.9870222.

Yaochu Jin. Surrogate-Assisted Evolutionary Computation: Recent Advances and Future Challenges. *Swarm and Evolutionary Computation*, 1 (2):61–70, 2011. ISSN 2210-6502. doi: https://doi.org/10.1016/j.swevo.2011.05.001. URL https://www.sciencedirect.com/science/article/pii/S2210650211000198.

Ian T. Jolliffe and Jorge Cadima. *Principal Component Analysis.* Springer, 2nd edition, 2016. ISBN 978-0-387-95442-4.

J. Kennedy and R. Eberhart. Particle Swarm Optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 39(6):1362–1381, 2009.

Arnaud Liefooghe, Sébastien Verel, Benjamin Lacroix, Alexandru-Ciprian Zăvoianu, and John McCall. Landscape features and Automated Algorithm Selection for Multi-Objective Interpolated Continuous Optimisation Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, page 421–429, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383509. doi: 10.1145/3449639.3459353. URL https://doi.org/10.1145/3449639.3459353.

Arnaud Liefooghe, Gabriela Ochoa, Sébastien Verel, and Bilel Derbel. Pareto Local Optimal Solutions Networks with Compression, Enhanced Visualization and Expressiveness. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 713–721, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701191. doi: 10.1145/3583131.3590474. URL https://doi.org/10.1145/3583131.3590474.

Katherine M Malan and Andries P Engelbrecht. A Survey on Fitness Landscapes and Landscape Analysis Techniques. *Swarm and Evolutionary Computation*, 61:100848, 2021.

Krzysztof Michalak. Low-Dimensional Euclidean Embedding for Visualization of Search Spaces in Combinatorial Optimization. *IEEE Transactions on Evolutionary Computation*, 23(2):232–246, 2019. doi: 10.1109/TEVC.2018.2846636.

Salifu Nanga, Ahmed Bawah, Benjamin Ansah, B. Mac-Issaka, Nii Odai, Samuel Kwaku Obeng, and Ampem Nsiah. Review of Dimension Reduction Methods. *Journal of Data Analysis and Information Processing*, 09:189–231, 01 2021. doi: 10.4236/jdaip.2021.93013.

Gabriela Ochoa, Katherine Malan, and Christian Blum. Search Trajectories Illuminated. *ACM SIGEVOlution*, 14:1–5, 07 2021a. doi: 10.1145/3477379.3477381.

Gabriela Ochoa, Katherine Malan, and Christian Blum. Search Trajectory Networks: A Tool for Analysing and Visualising the Behaviour of Metaheuristics. *Applied Soft Computing*, 109:107492, 05 2021b. doi: 10.1016/j.asoc.2021.107492.

Marcos Oliveira, Carmelo JA Bastos-Filho, and Ronaldo Menezes. Towards a Network-Based Approach to Analyze Particle Swarm Optimizers. In *2014 IEEE Symposium on Swarm Intelligence*, pages 1–8. IEEE, 2014.

Adam P. Piotrowski, Jaroslaw J. Napiorkowski, and Agnieszka E. Piotrowska. Population Size in Particle Swarm Optimization. *Swarm and Evolutionary Computation*, 58:100718, 2020. ISSN 2210-6502. doi: https://doi.org/10.1016/j.swevo.2020.100718. URL https://www.sciencedirect.com/science/article/pii/S2210650220303710.

Zbyněk Pitra, Jakub Repický, and Martin Holeňa. Landscape Analysis of Gaussian Process Surrogates for the Covariance Matrix Adaptation Evolution Strategy. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, page 691–699, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361118. doi: 10.1145/3321707.3321861. URL https://doi.org/10.1145/3321707.3321861.

Hartmut Pohlheim. Multidimensional Scaling for Evolutionary Algorithms—Visualization of the Path through Search Space and Solution Space Using Sammon Mapping. *Artif. Life*, 12(2):203–209, mar 2006. ISSN 1064-5462. doi: 10.1162/106454606776073305. URL https://doi.org/10.1162/106454606776073305.

Radka Poláková, Josef Tvrdík, and Petr Bujok. Differential Evolution with Adaptive Mechanism of Population Size According to Current Population Diversity. *Swarm and Evolutionary Computation*, 50:100519, 2019. ISSN 2210-6502. doi: https://doi.org/10.1016/j.swevo.2019.03.014. URL https://www.sciencedirect.com/science/article/pii/S2210650218301408.

Kenneth Price, Rainer M. Storn, and Jouni A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization.* Springer-Verlag, 2005.

Sam T. Roweis and Lawrence K. Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, 2000.

Shubham Sharma, Jette Henderson, and Joydeep Ghosh. CERTIFAI: Counterfactual Explanations for Robustness, Transparency, Interpretability, and Fairness of Artificial Intelligence Models. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, Feb 2020. doi: 10.1145/3375627.3375812. URL http://dx.doi.org/10.1145/3375627.3375812.

Manjinder Singh, Alexander E. I. Brownlee, and David Cairns. Towards Explainable Metaheuristic: Mining Surrogate Fitness Models for Importance of Variables. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, page 1785–1793, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450392686. doi: 10.1145/3520304.3533966. URL https://doi.org/10.1145/3520304.3533966.

Lenka Skanderova, Tomas Fabian, and Ivan Zelinka. Small-World Hidden in Differential Evolution. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3354–3361. IEEE, 2016.

Lydia Taw, Nishant Gurrapadi, Mariana Macedo, Marcos Oliveira, Diego Pinheiro, Carmelo Bastos-Filho, and Ronaldo Menezes. Characterizing the Social Interactions in the Artificial Bee Colony Algorithm. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 1243–1250. IEEE, 2019.

Josh Tenenbaum, V. de Silva, and John Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500): 2319–2323, 2000.

P Valkovič. Bbob torch. https://github.com/PatrikValkovic/BBOBtorch, 2021.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing Data Using t-SNE. *Journal of machine learning research*, 9(11), 2008.

Laurens Van Der Maaten, Eric Postma, Jaap Van den Herik, et al. Dimensionality Reduction: A Comparative Review. *J Mach Learn Res*, 10 (66-71):13, 2009.

Aidan Wallace, Alexander E. I. Brownlee, and David Cairns. Towards Explaining Metaheuristic Solution Quality by Data Mining Surrogate Fitness Models for Importance of Variables. In Max Bramer and Richard Ellis, editors, *Artificial Intelligence XXXVIII*, pages 58–72, Cham, 2021. Springer International Publishing. ISBN 978-3-030-91100-3.

Kilian Q Weinberger, Fei Sha, and Lawrence K Saul. Learning a Kernel Matrix for Nonlinear Dimensionality Reduction. In *Proceedings of the twenty-first international conference on Machine learning*, page 106, 2004.

Zhi-Hui Zhan, Jun Zhang, Yun Li, and Henry Shu-Hung Chung. Adaptive Particle Swarm Optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(6):1362–1381, 2009. doi: 10.1109/TSMCB.2009.2015956.

A TOP CONTRIBUTING VARIABLES - 25% FITNESS

Alg	Prob	Top 4 PC1	Top 4 PC2	Top 4 PC3
CMAES	F1	3, 4, 8, 5	7, 9, 10, 6	6, 9, 10, 1
DE	F1	3, 8, 4, 2	5, 2, 4, 3	2, 5, 8, 10
GA	F1	3, 8, 4, 2	2, 4, 5, 3	5, 2, 3, 10
PSO	F1	3, 4, 8, 5	2, 9, 5, 7	1, 9, 7, 5
CMAES	F2	8, 4, 5, 3	8, 9, 10, 7	7, 9, 8, 6
DE	F2	8, 4, 3, 5	8, 6, 5, 10	8, 10, 6, 5
GA	F2	8, 4, 3, 5	8, 5, 6, 2	8, 9, 6, 7
PSO	F2	4, 8, 3, 5	8, 10, 9, 5	8, 7, 9, 10
CMAES	F3	3, 4, 8, 2	1, 5, 2, 9	7, 6, 9, 1
DE	F3	8, 3, 4, 2	4, 8, 3, 5	8, 3, 4, 5
GA	F3	3, 8, 4, 2	8, 4, 3, 2	8, 3, 4, 2
PSO	F3	3, 4, 8, 5	8, 3, 4, 5	8, 4, 9, 5
CMAES	F4	3, 8, 4, 5	5, 3, 1, 9	6, 10, 3, 5
DE	F4	3, 8, 4, 2	3, 5, 8, 1	3, 5, 10, 2
GA	F4	8, 4, 2, 3	3, 8, 4, 5	3, 10, 4, 6
PSO	F4	3, 8, 4, 2	3, 5, 8, 4	3, 10, 5, 4
CMAES	F5	10, 8, 6, 9	9, 1, 10, 3	5, 4, 3, 7
DE	F5	9, 8, 10, 7	10, 9, 8, 7	10, 9, 8, 5
GA	F5	7, 8, 5, 6	9, 10, 1, 8	8, 6, 7, 2
PSO	F5	8, 6, 10, 7	10, 9, 2, 7	9, 10, 7, 8
CMAES	F6	4, 8, 5, 2	3, 5, 2, 4	3, 1, 2, 6
DE	F6	4, 8, 3, 2	3, 2, 5, 6	3, 2, 6, 5
GA	F6	4, 8, 10, 2	3, 2, 5, 6	3, 2, 5, 1
PSO	F6	4, 8, 5, 3	3, 2, 5, 4	3, 1, 7, 5

Table 3. First 25% F1-F6 Top 4 Cont. Vars

Alg	Prob	Top 4 PC1	Top 4 PC2	Top 4 PC3
CMAES	F7	3, 4, 8, 2	3, 4, 8, 5	3, 4, 8, 5
DE	F7	8, 3, 4, 2	8, 4, 3, 10	3, 4, 8, 5
GA	F7	3, 4, 8, 7	3, 4, 5, 8	3, 4, 8, 5
PSO	F7	4, 3, 8, 2	3, 8, 4, 5	3, 8, 4, 9
CMAES	F8	3, 4, 5, 8	3, 4, 8, 5	2, 3, 4, 8
DE	F8	4, 3, 5, 8	4, 3, 5, 9	3, 8, 4, 2
GA	F8	3, 4, 8, 5	3, 4, 8, 5	8, 3, 2, 4
PSO	F8	3, 4, 5, 9	3, 4, 5, 8	7, 3, 4, 9
CMAES	F9	2, 9, 7, 6	4, 2, 8, 9	8, 4, 7, 2
DE	F9	6, 1, 10, 9	7, 5, 6, 3	7, 5, 6, 1
GA	F9	10, 3, 9, 6	6, 5, 1, 3	6, 5, 7, 1
PSO	F9	3, 10, 9, 1	5, 1, 3, 9	1, 5, 6, 3
CMAES	F10	1, 2, 6, 3	1, 2, 9, 3	1, 2, 9, 4
DE	F10	1, 6, 4, 2	1, 2, 4, 8	1, 2, 9, 4
GA	F10	1, 2, 9, 6	1, 2, 9, 6	1, 9, 2, 8
PSO	F10	1, 9, 2, 6	1, 9, 6, 10	1, 9, 8, 10
CMAES	F11	6, 1, 2, 3	10, 9, 7, 3	7, 10, 9, 4
DE	F11	2, 6, 1, 8	2, 6, 1, 9	6, 1, 2, 9
GA	F11	2, 3, 5, 1	3, 2, 8, 6	10, 9, 2, 8
PSO	F11	8, 6, 1, 3	9, 1, 8, 4	3, 4, 6, 9
CMAES	F12	1, 2, 6, 3	3, 9, 1, 7	8, 5, 9, 10
DE	F12	6, 1, 3, 2	2, 6, 1, 9	8, 6, 2, 1
GA	F12	6, 2, 1, 3	6, 1, 2, 3	6, 1, 3, 2
PSO	F12	1, 2, 6, 3	3, 6, 9, 2	7, 3, 9, 10

Table 4. First 25% F7-F12 Top 4 Cont. Vars

Alg	Prob	Top 4 PC1	Top 4 PC2	Top 4 PC3
CMAES	F13	1, 6, 2, 3	5, 6, 10, 9	8, 4, 7, 5
DE	F13	6, 1, 2, 3	2, 3, 1, 4	2, 1, 4, 8
GA	F13	6, 1, 2, 3	2, 1, 6, 3	2, 4, 6, 10
PSO	F13	1, 2, 6, 3	1, 9, 2, 6	9, 2, 7, 5
CMAES	F14	1, 2, 6, 3	9, 8, 1, 2	10, 9, 2, 7
DE	F14	6, 1, 2, 3	2, 1, 8, 3	2, 1, 3, 4
GA	F14	6, 1, 2, 3	1, 2, 3, 8	2, 1, 3, 4
PSO	F14	1, 2, 6, 3	1, 2, 6, 9	9, 7, 3, 5
CMAES	F15	1, 2, 6, 3	9, 2, 6, 3	9, 7, 10, 4
DE	F15	1, 6, 2, 3	1, 6, 2, 3	1, 2, 6, 3
GA	F15	1, 2, 6, 3	2, 1, 6, 3	2, 6, 1, 3
PSO	F15	1, 2, 6, 3	6, 2, 1, 9	6, 9, 1, 2
CMAES	F16	6, 8, 7, 9	10, 7, 5, 3	5, 7, 10, 3
DE	F16	9, 3, 10, 2	10, 1, 9, 2	3, 8, 9, 10
GA	F16	4, 5, 2, 1	5, 7, 10, 4	8, 6, 9, 10
PSO	F16	9, 2, 8, 5	9, 7, 5, 2	5, 9, 10, 1
CMAES	F17	1, 2, 6, 3	7, 5, 8, 9	5, 9, 10, 8
DE	F17	6, 1, 2, 3	6, 2, 1, 3	2, 1, 3, 6
GA	F17	6, 2, 1, 3	6, 1, 2, 3	1, 2, 3, 6
PSO	F17	1, 2, 6, 3	1, 2, 3, 6	1, 2, 6, 3
CMAES	F18	1, 2, 6, 3	9, 2, 3, 4	7, 10, 9, 8
DE	F18	6, 1, 3, 2	2, 1, 6, 3	2, 6, 1, 3
GA	F18	6, 1, 2, 3	6, 2, 1, 9	2, 3, 6, 1
PSO	F18	2, 1, 6, 3	2, 6, 9, 1	9, 6, 3, 2

Table 5. First 25% F3-F18 Top 4 Cont. Vars

Alg	Prob	Top 4 PC1	Top 4 PC2	Top 4 PC3
CMAES	F19	8, 6, 9, 10	6, 4, 2, 5	6, 10, 7, 1
DE	F19	9, 4, 1, 8	7, 5, 1, 6	5, 7, 2, 8
GA	F19	9, 4, 8, 2	7, 2, 4, 8	6, 7, 4, 5
PSO	F19	9, 2, 10, 3	6, 9, 3, 2	4, 8, 2, 3
CMAES	F20	10, 6, 8, 7	10, 8, 6, 7	10, 9, 8, 7
DE	F20	6, 7, 9, 8	7, 8, 9, 10	10, 8, 9, 6
GA	F20	7, 3, 2, 4	7, 3, 4, 6	7, 4, 3, 5
PSO	F20	10, 8, 6, 1	8, 10, 6, 1	8, 6, 10, 4
CMAES	F21	2, 5, 8, 4	7, 1, 9, 3	6, 9, 7, 3
DE	F21	2, 4, 8, 5	4, 2, 5, 8	4, 5, 2, 8
GA	F21	2, 5, 8, 4	8, 5, 2, 4	2, 4, 10, 5
PSO	F21	8, 2, 4, 5	8, 5, 4, 3	5, 8, 9, 6
CMAES	F22	3, 5, 4, 8	10, 7, 9, 5	1, 8, 4, 9
DE	F22	3, 1, 5, 2	5, 1, 6, 7	7, 5, 9, 1
GA	F22	3, 9, 5, 6	5, 3, 9, 6	9, 8, 5, 6
PSO	F22	3, 9, 5, 6	5, 3, 10, 6	4, 7, 10, 2
CMAES	F23	5, 1, 7, 10	1, 4, 9, 7	2, 5, 7, 4
DE	F23	6, 2, 7, 5	2, 10, 8, 6	10, 6, 1, 2
GA	F23	9, 8, 7, 10	10, 5, 9, 4	2, 5, 9, 3
PSO	F23	6, 5, 9, 3	10, 5, 4, 1	4, 7, 10, 2
CMAES	F24	4, 2, 9, 5	8, 3, 10, 1	10, 8, 6, 1
DE	F24	6, 4, 1, 7	1, 3, 2, 5	6, 8, 3, 10
GA	F24	9, 1, 4, 6	7, 3, 4, 1	9, 6, 2, 4
PSO	F24	9, 4, 5, 2	9, 2, 4, 5	9, 2, 4, 5

Table 6. First 25% F19-F24 Top 4 Cont. Vars