FYVIE, M., MCCALL, J.A.W., CHRISTIE, L.A. and BROWNLEE, A.E.I. 2023. Explaining a staff rostering genetic algorithm using sensitivity analysis and trajectory analysis. In GECCO'23 companion: proceedings of the 2023 Genetic and evolutionary computation conference companion, 15-19 July 2023, Lisbon, Portugal. New York: ACM [online], pages 1648-1656. Available from: https://doi.org/10.1145/3583133.3596353

Explaining a staff rostering genetic algorithm using sensitivity analysis and trajectory analysis.

FYVIE, M., MCCALL, J.A.W., CHRISTIE, L.A. and BROWNLEE, A.E.I.

2023

Explaining a Staff Rostering Genetic Algorithm using Sensitivity Analysis and Trajectory Analysis.

Martin Fyvie
m.fyvie@rgu.ac.uk
Robert Gordon University
Aberdeen, Scotland

John A. W. McCall
j.mccall@rgu.ac.uk
Robert Gordon University
Aberdeen, Scotland

Lee A. Christie
l.a.christie@rgu.ac.uk
Robert Gordon University
Aberdeen, Scotland

Alexander E.I. Brownlee
alexander.brownlee@stir.ac.uk
University of Stirling
Stirling, Scotland

ABSTRACT

In the field of Explainable AI, population-based search metaheuristics are of growing interest as they become more widely used in critical applications. The ability to relate key information regarding algorithm behaviour and drivers of solution quality to an end-user is vital. This paper investigates a novel method of explanatory feature extraction based on analysis of the search trajectory and compares the results to those of sensitivity analysis using "Weighted Ranked Biased Overlap". We apply these techniques to search trajectories generated by a genetic algorithm as it solves a staff rostering problem. We show that there is a significant overlap between these two explainability methods when identifying subsets of rostered workers whose allocations are responsible for large portions of fitness change in an optimization run. Both methods identify similar patterns in sensitivity, but our method also draws out additional information. As the search progresses, the techniques reveal how individual workers increase or decrease in the influence on the overall rostering solution's quality. Our method also helps identify workers with a lower impact on overall solution fitness and at what stage in the search these individuals can be considered highly flexible in their roster assignment.

CCS CONCEPTS

• **Theory of computation** → **Models of computation**; • **Computing methodologies** → **Search methodologies**; **Genetic algorithms**.

KEYWORDS

Evolutionary Algorithms, Principal Component Analysis, Algorithm Trajectories, Sensitivity Analysis, Explainable AI (XAI)

ACM Reference Format:
Martin Fyvie, John A. W. McCall, Lee A. Christie, and Alexander E.I. Brownlee. 2023. Explaining a Staff Rostering Genetic Algorithm using Sensitivity Analysis and Trajectory Analysis.. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3583133.3596353

1 INTRODUCTION

As the application of Evolutionary Algorithms (EAs) and other population-based search metaheuristics gains popularity in domains that involve cooperation with end-users, it becomes increasingly important to instil trust and acceptance of these techniques. One approach to achieving this is through the application of Explainable Artificial Intelligence (XAI) techniques, which has experienced an increase of interest in recent years, likely driven by the growing use of case-based reasoning and deep learning decision-making strategies in end-user-facing solutions.

As shown in the 2019 survey of the field of XAI [1], EAs have been used in a variety of ways to improve and enhance the explanation generation process. Examples of their use can be seen in the creation of counterfactuals [2] and as a refinement method for fuzzy rule generation [3]. Despite the broad scope of XAI as seen in [1], EAs as a focus of XAI studies have been under-represented. However, this is changing as XAI's popularity grows, as demonstrated in the 2022 GECCO XAI workshop and the 2023 EvoStar conference [4].

There exists a wide array of approaches to the generation of explanations regarding AI-generated solutions and their decision-making processes. Examples of model-specific explanation methods include those applied to Multi-Layer Neural Networks (MLNNs) [5] and Convolutional Neural Networks (CNNs) [6]. Sensitivity and Saliency Mapping can be used to create a form of visual explanation regarding algorithm decisions [7].

A further approach is to extract an understanding from the sensitivities of the fitness function to specific variables. Similarly, the use of surrogate models [8] can achieve a similar level of explanation through the mining of feature importance from these explicit models of the fitness function [9, 10]. Other relevant work in Landscape Analysis is using Search Trajectory Networks [11, 12] to visualise algorithm search behaviour.

Sensitivity Analysis (SA), an interesting application of which can be seen in [13], can identify the sensitivity to fitness that a variable has. Further examples of feature sensitivity [14, 15] have been used as a model-agnostic approach.

In this paper, we present our approach to feature extraction from the search trajectories of EAs, known as Trajectory Analysis (TA), which was first proposed in [16]. These features are represented as

variable subsets of high relevance to an end user. We decompose the search trajectories of an EA using Principal Component Analysis (PCA) techniques and subsequently mine them for pertinent and explanatory features. For comparison, we examine the results obtained by applying Sensitivity Analysis (SA), a method that employs random samples from search trajectories to identify variables of interest. We showcase the first application of TA to a real-world problem.

The remainder of this paper is structured as follows: Section 2 provides background information on our definition of trajectory analysis, the proposed method of explainable feature extraction, sensitivity analysis, and the similarity scoring method used to compare our results. Section 3 describes how our algorithm trajectory datasets are generated and organised for analysis. Section 4 presents the comparative experimental results between TA and SA on the generated datasets, while Section 5 summarises the work and presents our conclusions.

2 EXPLAINABILITY METHODS

We begin with a definition of search trajectories and our approach to analysing them to generate explanations. This is followed by a brief summary of the well-established techniques of sensitivity analysis and how they might be used to explain the progress of an EA. SA provides our baseline for comparison. Lastly we introduce the ranking method we use for the comparison between the SA results and our own and how we define the convergence of a variable.

2.1 Search Trajectory Definition

In this paper, a search trajectory, T, is defined as a collection, X, of solutions visited during a population-based search, ordered by their generation g (Equation 1). Each solution's variables will be drawn from the discrete space Z. This represents a list of $\mathbb{Z}^{gNn}$ solutions ordered by generation g, pop size N and problem dimension n. For this paper all solution variables are integers.

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

The search trajectories in our study were created by running a genetic algorithm on the target problem (fully defined in Section 3.1) 100 times, each using a random seed to create the initial population of solutions. Each run has 50 generations (g) of 10 solutions. Each solution (x) consisted of a discreet variable string encoding with a fixed length of $n = 141$. The encoding employed for these solutions was in the integer domain $\mathbb{Z}$ as outlined earlier. These values were selected after a period of refinement to allow for sufficient run length. It is important to note that this definition of a trajectory is not limited to the integer domain. The approach can also be applied to problems in which the solutions lie in real-valued space $\mathbb{R}$ however the problem definition determined the discreet value solution structure and solution length.

2.2 Search Trajectory Analysis

The Search Trajectory Analysis approach outlined in this paper requires the pre-processing of each optimization runs trajectory (T). To achieve this, Principal Component Analysis (PCA) was used to decompose these datasets. Once this process is complete our feature extraction method can be applied. Preliminary testing of the results had shown high covariance between some of the variables in the solution populations. Because of this, the application of PCA was applied via the use of Single Value Decomposition (SVD) of the correlation matrix generated from each trajectory. The resulting directional vectors from using PCA on our dataset of trajectories (Equation 2) are a set of m, $n \times 1$ orthonormal eigenvectors in $\mathbb{R}^n$. The p^i vectors, also known as Principal Components, are comprised of the elements $[p_1^i, \ldots, p_n^i]$ which represent the weighting of each variable. These weightings or coefficients describe the contribution of each variable to the corresponding principal component. The absolute value of these coefficients indicates the variable's influence in maximizing variance across the dataset through the best-fit hyperplane. The resulting subspace characterizes the algorithm's search trajectory in terms of variable variance and how it changes with the algorithm's position on the fitness gradient.

$$\begin{aligned} P &= [p^1, \ldots, p^m]^\intercal, m \leq n \\ p^i &= [p_1^i, \ldots, p_n^i] \end{aligned} \tag{2}$$

Our feature extraction method is based on the concept of variable contributions shown in Equation 3. These values are used to calculate the overall contribution each variable has in determining a solution's position in the search space at a given generation.

$$C_i' = \sum_{k=1}^{N} \left(\frac{\left(\sum_{j=1}^{n} p_j^i x_j^k \right)}{N} \right) p^i. \tag{3}$$

Here, C_i' belonging to $\mathbb{Z}$, is a vector of mean variable contributions across component p^i of any given generation of size N solutions with dimension n. In Equation 3, for any given generation of solutions, the contribution of each variable (j) in a solution (k) is calculated as the product of the variable's value and its corresponding component coefficient in principal component i. The mean value across all solutions in that generation is taken and stored in C_i'. Each instance of C_i' represents the contribution of the variables across component i at generation k. This provides us with a method for describing the influence each variable in the problem has in determining the current population's overall position.

For each optimisation run we collect the resulting contribution vectors at each generation. This provides us with a generation-by-generation view of the mean contribution value that each variable is calculated to have. Outlined fully in Section 3.3 is a method of determining which generations were grouped together for our analysis. For each collection of generations used, we then ordered them by ranking the mean absolute value of the contribution of each variable from 1 to n. Here 1 denoted the most influential and n denoted the least. The final ranking of each variable is determined by taking the mean value across all 100 runs at each generation.

2.3 Sensitivity Analysis

Sensitivity Analysis (SA) [17] can be used to measure the sensitivity of the fitness function to a variable by its inclusion or exclusion from a linear regression on the fitness values. This provides an insight into how sensitive a solution's fitness is to the presence

of that variable. There are three main methods for applying SA to a dataset in terms of variable selection, these being Forwards Selection, Backward Selection, and Bi-Directional Elimination [18]. Forward selection involves the addition of variables one at a time, beginning with the variable having the highest correlation with the dependent variable. This is repeated until no further significant improvement is noticed. Backward selection involves creating a model with all possible variables present. The significance of each variable is calculated and the least significant is removed. This is repeated until no further improvement is noticed. Lastly, Bi-Directional elimination draws from both previous approaches to add and remove variables based on their calculated significance value. Inspired by the work in [13], our method also utilizes a step-wise linear regression analysis of the rank-transformed variable values. A major difference in our approach was to use backward selection as the total number of variables ($n = 141$) was significantly smaller than the total sample size in each trajectory. This also helps with dealing with variables with high collinearity as backward selection is more likely to retain them when compared to forward or bi-directional selection which may remove them all. Variables are removed from the selection pool using least square regression. This is done should the corresponding prediction p-value be greater than the significance level which was set at $p = 0.05$. The results of this process are a set of standardised rank regression coefficients (SSRC) representing the overall impact each variable has on the fitness value of a solution. We take these coefficients and apply a further ranking to them so that the output is a set of ranked variables based on their SSRC. This process can be seen in a pseudo-code form in Algorithm 1

Algorithm 1 Sensitivity Analysis with Backward Selection

Define a significance level $\alpha = 0.05$
Perform linear regression with all variables included
while there are variables in the model **do**
    Calculate p-values for all variables in the model
    Find the variable with the highest p-value: p_{max}
    if $p_{max} > \alpha$ **then**
        Remove the variable with the highest p-value from the model
    else
        Break the loop
    end if
end while
Calculate standardized rank regression coefficients (SSRC) for remaining variables
Rank the variables based on their SSRC

2.4 Weighted Rank Biased Overlap

Weighted Rank Biased Overlap (WRBO) is a similarity measure that allows the comparison of two lists of ranked variables [19]. The method uses an "Average Overlap" weight parameter which can add additional weight to a selection of the highest-ranked members of these lists. The process generates a score that takes a value of $[0, 1]$, where 0 shows no overlap or similarity and 1 shows a complete overlap and full similarity between two ranked lists both in terms of membership and rank order. This process can be seen in Algorithm 2 which was created from a Python implementation of the WRBO function outlined in [20].

Algorithm 2 Rank Biased Overlap (RBO) Python

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

Recent work involving the use of WRBO [21] highlights its ability to increase the interpretability of Machine Learning models. Classically, methods such as Spearman Rank Correlation or Kendall Tau would be used for the comparison of ranked lists. A similar analysis using Spearman Rank Correlation was attempted however the results did not clearly identify patterns in the data. One advantage that WRBO has over these alternative approaches is that they both require lists of ranked items to contain the same number of elements and the same elements themselves. This first issue does not occur when comparing the rankings of all 141 variables however they are unable to perform this task reliably on the top 10 lists. This is because there is no guarantee that both lists will contain the same 10 variables.

The WRBO process is highly relevant to our study as it allows us to place a higher emphasis on high-ranking values when comparing the results between the PCA approach and SA. Variables placed nearer the top of each list will have a higher mean ranking and so will be considered of higher importance or sensitivity. After some initial testing, a $WRBO - p$ value of 0.9 was selected. This value determines the contribution of the top variables in each list. With a value of 0.9, the top 10 variables were responsible for 85.56% to the total scoring. This value also aligned with our perception that variable subsets of size greater than 10 begin to become difficult to interpret if presented to an end-user.

2.5 Variable Convergence

In order to gain insights regarding variable convergence we calculated, for each variable, the generation at which its variance had dropped below a selected threshold. To achieve this we calculated the within-generation variance using Equation 4 for each variable, in each generation of every run.

$$Var_{ij} = \frac{1}{N-1} \sum_{k=1}^{N} (x_{ijk} - \bar{x}_{ij})^2 \tag{4}$$

Here, we calculate the variance of the i-th variable of the k-th solution in population j. N is the total population size as outlined in Equation 1. $\bar{x}_{ij}$ is the mean value of the i-th variable in that solution.

For each optimization run, this generates a list showing the variance of each variable at each generation. For each variable, we normalize these values across all generations and then sort them in descending order. In this paper, we consider a variable to have mostly converged when the normalized variance value has reached or is below the threshold of 0.01 or 1% of the total observed variance.

3 TRAJECTORY DATASET GENERATION

3.1 Rostering Problem

The problem that forms the focus of our case study is a simplified variant of the staff roster allocation problem described in [22, 23]. This problem aims to minimize the overall range between the minimum and the maximum number of workers allocated on each day of the week. This ensures a consistent number of employees working on each day of the week across a two-week period. Rosters for each worker were created indicating shift start and end times of varying lengths representing up to a three-month period, ensuring at least two consecutive days off each week. If an allocated roster was shorter than the two-week scheduling period, a worker would repeat that pattern. Each worker was assigned a randomly selected subset of these rosters with a minimum of 1 and a maximum of 5 options. For this problem, the indices of the rosters in the subsets are treated as the worker's preference. This means that the roster at index 1 is the worker's preferred choice. Index two would be their second preference and so on. An example solution representation can be seen in Table 1. Here, each worker is represented by the variable x. The value assigned to them is the index of the roster they have currently been assigned to from their specific subset, e.g., a value of 3 would indicate that worker x_2 has been assigned to their third choice of roster.

Table 1: Cost Function Weight (W) Effectiveness

x_1	x_2	x_3	...	x_n
1	3	2	...	2

In the original problem, $A_{i,j}$ denotes the "attendance matrix" of size $n \times 7$, with rows representing weeks and columns representing days. This maps to our trajectory definition in that each value in A is the summation of the number of workers scheduled to work on that specific day j and week i, based on whether their current assigned roster has scheduled them to work.

The range is calculated by taking the minimum and maximum of matrix $\mathbf{A}$ for each column j, used to compute the normalised range of column j as shown in Equation 5.

$$R_j = \frac{\max_j \left(a_{ij}\right) - \min_j \left(a_{ij}\right)}{\max_j \left(a_{ij}\right)} \tag{5}$$

The cost function aims to minimize the overall relative factor across all days, squared to smooth out range values, with a set of weights w to reduce the impact of weekends with lower resources as shown in Equation 6. A set of weights, w, are included in which the weight w_j is a weighting applied to day j. This is done to reduce the overall impact of the weekends ($j = 6, 7$) having significantly lower levels of available resources, resulting in a higher relative factor that could disproportionately affect the score. A daily weighting vector of $w = (1, 1, 1, 1, 1, 10, 10)$ was used as in the original problem. As noted in that paper "a range of 10 on Saturday should not be considered the same as a range of 10 on any other day of the week due to the smaller number of attending resources". The higher weight on weekends is designed to reduce this impact.

$$cost = \sum_{1 \le j \le 7} w_j R_j^2 + \left(\sum_{i \in X} P_i \right) W \tag{6}$$

The cost function is modified to include X, the set of all workers, and $P = (p_{i,j})$, a binary array in which $p_i = 1$ if worker i has been allocated their first preference of roster, otherwise 0. The second summand in the cost function sums the total number of workers not allocated their first preference with overall weight, W, applied to this soft constraint. This weight value was set at 0.001. The value was selected after testing to have minimal impact on early, high-fitness solutions. As the range is reduced, the effect W has on a solution's fitness increases. Table 2 shows how this weight is used to increase the fitness of a solution when compared to a similar solution with the same range value. The percentage change in fitness as the overall range is reduced is shown in the "W %" column. Shown in bold is the solution that benefits most from this weighting when compared to similar solutions.

The aim of this addition is to introduce some bias in the search towards solutions with a lower number of workers that did not get assigned their first, preferred roster in the solution. This occurs mostly when the range has been significantly reduced.

Table 2: Cost Function Weight (W) Effectiveness

Fitness	Range	Tot. Non-1st Choice	W %
0.464	**0.397**	**67**	**16.876**
0.465	0.397	68	17.128

3.2 Genetic Algorithm Trajectory Creation

The EA implementation used in this paper was that of a $(\mu + \lambda)$ Genetic Algorithm (GA), 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. The solutions used a discrete variable string encoding so that each worker was represented by an integer denoting the index of the roster they were currently assigned to out of their possible choices. This means for a given worker, a value of 5 would represent the 5th preference of roster. As each worker could have a minimum of 1 and a maximum of 5 possible choices, the crossover and mutation operators selected allow for the potential to repair a solution. As these operators may assign a worker with 3 possible rosters an index greater than 3 for example, these methods contain repair mechanisms capable of

dealing with this situation. The implementation of these operators are taken from the Python Multi-Objective Optimisation (PYMOO) library [24]. Tournament Selection and Simulated Binary Crossover were selected as they were the default used in examples provided in PYMOO's documentation. Refinement and parameter tuning was not performed as the exact performance of the GA was not the focus of this study. After some initial testing, the GA's performance was deemed acceptable in that it reliably found better fitness solutions over the course of the optimization run, creating suitable search trajectories for our purpose. Mutation was handled by the PYMOO implementation of Polynomial Mutation, details of which can be found in [25].

3.3 Fitness Quartiles

In order to gain insight into which variables are driving the optimization process at different stages of the search, the trajectories were split into four "Fitness Quartiles". These were calculated by measuring the mean solution fitness in each generation of an optimization run. This was used to select the upper and lower bounds, in terms of generation number, for an approximately 25% share of total fitness change. This meant that each quartile would represent approximately one-quarter of all fitness change. An illustrative example of this can be seen in Figure 1.

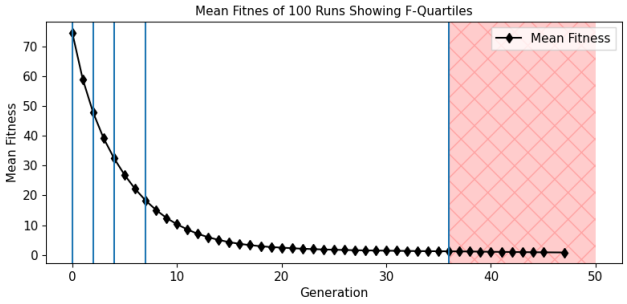

Figure 1: Mean Fitness and Fitness Quartiles

Here, each vertical line represents an example of the upper and lower bounds of the quartiles, staring at generation 0. Generations between these bounds are used in their corresponding quartile for our analysis. This provides us with four separate stages in a search trajectory in which to apply our methods and compare which variables were seen to be driving fitness change more than others. Each optimization run had these upper and lower bounds calculated individually so that solutions captured in each quartile were specific to each run. As seen in Figure 1 there is the potential for a long "tail" of low-variance solutions when the algorithm is close to convergence. To avoid these from having a disproportionate effect on the calculation of the fitness windows and variable importances in each, we assign the fourth quartile at the generation at which 99% of fitness has been achieved rather than 100%. The figure shows how using the 99% value truncates the trajectory, removing these low-variance solutions from the analysis.

4 EXPLAINABILITY RESULTS AND COMPARISONS

The purpose of our experimentation is to discover whether our proposed method is capable of identifying variables of high importance at differing stages of an optimization run. We use the variable subsets identified as being of high sensitivity from the results of the Sensitivity Analysis and compare the Trajectory Analysis results to these. This offers an insight into whether our proposed method is discovering similar features as the SA approach as well as highlighting potential variable importances not seen in the SA results.

Figure 2 illustrates the distribution of generations in which each variable was calculated to have converged across all 100 runs in our dataset. This is measured by observing the generation in which the variance of that variable reaches less than 1% across all solutions in a generation. These have been ordered from earliest to latest to converge from left to right. The red horizontal lines indicate the mean generation for the fitness quartiles as seen in Table 3 and help identify in which quartile a variable tends to converge.

Table 3: Mean Fitness Quartile Generations

Q	Start	End	Total Gens	Fitness %
1	0	2	2	0 - 25
2	2	4	2	25 - 50
3	4	7	3	50 - 75
4	7	36	29	75 - 99

This figure also highlights that there are some variables that consistently converge earlier in the trajectory than other, regardless of the starting point that the search algorithms has.

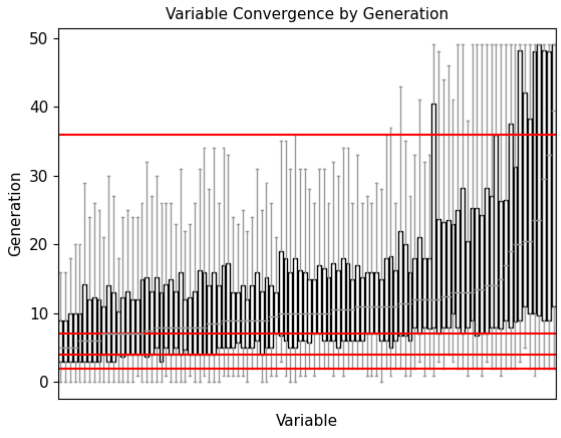

Figure 2: Variable Convergence by Generation - Ordered

Shown in Table 4 are the mean convergence generation numbers of the fastest and slowest to converge variables. Here we show that variables **{6,125,86,17,139}** all have a mean convergence generation

less than 9, placing these in the early stages of the 4th fitness quartile. This would place these variables at the far left of Figure 2.

Table 4: Mean Variable Convergence Generations

	First 5 Converging				
Variable	6	125	86	17	139
Mean Conv. Gen	6.16	6.38	7.18	7.29	8.21
	Last 5 Converging				
Variable	110	30	48	22	58
Mean Conv. Gen	26.78	27.74	27.79	28.25	30.32

Variables **{110,30,48,22,58}** are the last values to converge when measured by their mean convergence generation value. These would be placed on the far right of Figure 2.

4.1 Sensitivity Analysis Results

The sensitivity analysis results seen in Figure 3 shows the R-Squared values of the linear models used in the SA approach on the 100 simulation runs. Figure 3 shows the distribution of R-Squared values when the SA is applied to a subset of each simulation, defined by the fitness quartiles.

The results show that the approach produced R-Squared values above the threshold of 0.9 however the fitness quartiles do show some level of variation in the R-Squared values. The results show that as the search progresses, we see a slight decline in the R-Squared values as the model is able to explain slightly less variation in fitness, given the input variables during those quartiles.

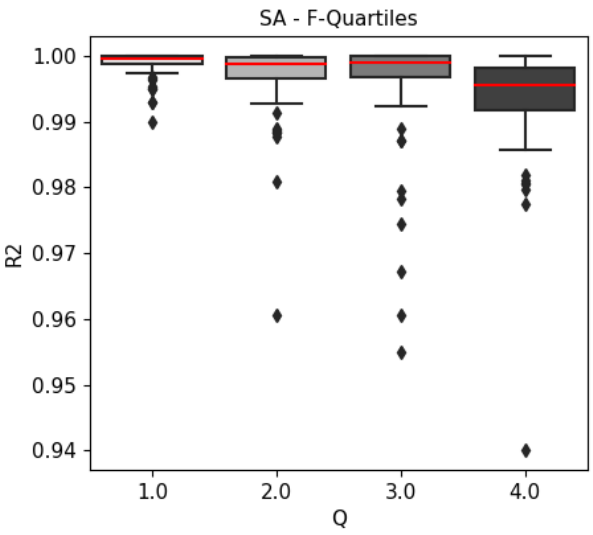

Figure 3: SA R-Squared Results Over 100 Runs Comparison

The variable rankings produced by the SA technique when applied to subsets of each optimization run by the fitness quartiles are shown in Table 5. Each row presents the top 10 variable indices when ranked from most to least sensitive for each fitness quartile. From this table, an example of changing variable sensitivity can be seen in variables {**98**, **17**} which were highly ranked in fitness quartiles Q1 and Q2. By Q3, only variable {**98**} continued to be highly ranked as {**17**} falls below the top 10 ranks.

Table 5: Sensitivity Analysis - Top 10 Ranked Variables by Quartile

Rank	1	2	3	4	5	6	7	8	9	10
Q1 Var	138	17	98	18	34	57	7	88	120	134
Q2 Var	17	98	135	32	46	48	12	70	41	76
Q3 Var	65	70	10	110	25	98	46	76	32	45
Q4 Var	68	45	46	98	85	70	27	42	113	18

The results show that variables {**98**, **17**} were highly ranked in Q1 and Q2 and {**98**} remained highly ranked in Q3 and Q4. Variable {**46**} was highly ranked in Q2, Q3 and Q4. Each fitness quartile shows some overlap with the others as well as identifying variables unique to that portion of the trajectory, indicating that as fitness is reduced, some variables remain highly sensitive while others vary considerably. Examples of variables that are only highly sensitive in one quartile include variables {**57**, **7**, **88**} in Q1 and {**85**, **66**, **125**} in Q4.

4.2 Search Trajectory Analysis Results

Figure 4 shows the percentage of explained variance by each component after the trajectories have been decomposed via PCA. It can be seen from this figure that there is a sharp drop-off of explained variance from component 2 onwards. Due to this rapid drop-off, the rest of the results are calculated using the first 3 components which are able to explain 45.86% of the variance.

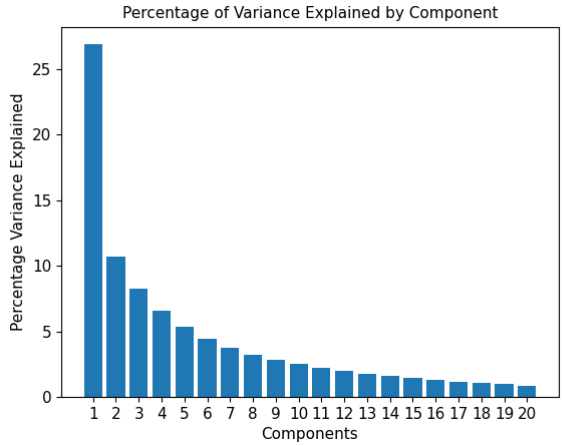

Figure 4: Explained Variance Percentage by Component

This rapidly reducing level of variance explained suggests that little would be gained by adding additional components to the

calculations versus the trade-off between higher explained variance and interpretability.

Table 6 presents the variable rankings produced by TA across all fitness quartiles. For each fitness quartile, the results are shown for each component. The results show that the ranking of variables based on contribution scores changes over the course of the trajectory, similar to what we see in the SA results.

Table 6: TA Variable Ranks - Fitness Quartiles

Quartile 1

Rank	1	2	3	4	5	6	7	8	9	10
PC1	6	17	112	34	98	71	2	32	138	95
PC2	98	17	2	112	6	138	127	32	34	62
PC3	138	6	98	17	34	95	139	112	125	126

Quartile 2

Rank	1	2	3	4	5	6	7	8	9	10
PC1	32	2	8	88	108	112	98	138	135	95
PC2	2	98	32	91	94	138	117	46	27	41
PC3	138	25	95	98	75	46	10	108	32	65

Quartile 3

Rank	1	2	3	4	5	6	7	8	9	10
PC1	2	32	8	138	91	38	88	25	98	65
PC2	91	2	94	41	32	133	27	98	117	38
PC3	138	75	25	91	10	65	46	32	27	44

Quartile 4

Rank	1	2	3	4	5	6	7	8	9	10
PC1	2	65	135	32	38	25	88	76	42	59
PC2	133	117	2	94	38	91	113	96	32	48
PC3	65	75	25	138	133	48	42	10	91	113

Variable {**98**} is ranked 5th, 1st and 3rd most contributing in Q1 across components PC1, PC2 and PC3 respectively. Over the course of optimizations, these rankings steadily drop between fitness quartiles until it no longer ranks in the top 10, suggesting a falling level of contribution over time. The opposite behaviour is seen in variable {**2**} which initially ranks 7th and 3rd across components PC1 and PC2 but rises to 1st and 2nd rank by Q3 and remains highly influential in these components for the remainder of the optimization. The mean convergence generations for these variables are 10.81 for {**98**} and 13.43 for {**2**}, showing that variable {**2**}, which converges on average much later than variable {**98**} is considered highly influential during the later stages of the optimization runs.

4.3 Sensitivity Analysis and Trajectory Analysis Results Comparison

To enable the comparison between the TA and SA ranking results, we apply the WRBO technique to generate similarity scores between the ranked lists. The WRBO score takes a value of [0,1] with 1 representing a perfect match in both membership and ranking order and 0 signifying no similarity at all. We used a weight parameter of $p = 0.9$ so that the top 10 variables contribute 85.56% towards the scores and the remaining variables contributed 14.44%. Shown in Table 7 is an example of the variable indices found to be in the top 10 rankings based on either sensitivity (SA) or contribution (TA). Values in bold are variables belonging to both lists.

Table 7: Quartile 1 Illustrative Result Comparison

Rank	1	2	3	4	5	6	7	8	9	10
PC1 Top 10	6	**17**	112	**34**	**98**	71	2	32	**138**	95
SA Top 10	**138**	**17**	**98**	18	**34**	57	7	88	120	134

Table 8 displays the results of the WRBO similarity scoring between the SA variable rankings and the TA results for each fitness quartile. The Table also highlights the overlap between the two method's top 10 ranked variables. The order of the variables in the "Shared - Top10" section is the order in which they are found in the TA results, from highest ranked to lowest. The results show that the earlier quartiles have a higher level of overlap between the SA results and the TA results. This overlap shows that both methods are identifying some of the same workers as being critical to the rostering plan. This overlap drops off quickly between quartiles Q3 and Q4 with the ranked list overlap falling from a maximum of 5 out of 10 to only 2 out of 10 across all three components. This indicates that the rank order of the variables is diverging over time between the two approaches.

Table 8: SA and TA WRBO and Variable Overlap Results

Quartile 1

	PC1	PC2	PC3
Shared - Top10	**{17,34,98,138}**	**{98,17,138,34}**	**{138,98,17,34}**
WRBO - Top10	0.374	0.406	0.574
WRBO - All	0.358	0.358	0.566

Quartile 2

	PC1	PC2	PC3
Shared - Top10	**{32,98,135}**	**{98,32,46,41}**	**{98,46,32}**
WRBO - Top10	0.199	0.357	0.209
WRBO - All	0.229	0.229	0.246

Quartile 3

	PC1	PC2	PC3
Shared - Top10	**{32,25,98,65}**	**{32,98}**	**{25,10,65,46,32}**
WRBO - Top10	0.175	0.093	0.328
WRBO - All	0.184	0.184	0.334

Quartile 4

	PC1	PC2	PC3
Shared - Top10	**{42}**	**{113}**	**{42,113}**
WRBO - Top10	0.044	0.044	0.088
WRBO - All	0.102	0.102	0.134

The similarity scores between the SA rankings and the first three components rankings in quartile 1 show that PC3 has the highest

similarity both across the top 10 variables and across all 141 at 0.574 and 0.566 respectively. All three components share the same four variables, **{17,34,98,138}**, in common with the SA rankings. As the top 10 variables are contributing approximately 85% towards this value, this would suggest that the differing WRBO scores between the components are due to the rank ordering of the remaining 131 variables. with component PC3 having overall closer rankings to the SA approach than the other components, its WRBO score is higher.

The results show that as the search continues, we see that the SA and TA variable rankings continue to diverge. When considering the scores for the top 10 variables, Quartile 1's highest similarity score is 0.574 in PC3. This reduces to 0.357 in PC2 during quartile 2. In quartile 3 this value decreases to 0.328 in PC3. By quartile 4 the largest drop in score is seen as the maximum value is 0.088 in PC3.

The WRBO scores across all 141 variables closely match those of the top 10 lists. Interestingly, there are occurrences of the WRBO score for all 141 variables being higher than the top 10 scores, despite their approximately 15% contribution to the value. An example of this can be seen in PC1 during quartiles Q2, Q3 and Q4. This implies that, due to the weight parameter, the higher WRBO-All score is due to a larger number of the remaining 131 variables being ranked similarly to the SA results. This causes the score to be higher than when only considering the top 10 variables.

This shows that while both approaches differ in their ranking of the most influential variables, lower influence variables are being both scored and ordered in a more similar fashion.

Lastly, Figure 5 shows the mean number of roster options available for each variable to take when filtered to the Top-10 ranked variables of each component and SA. These results show that over the course of the optimization, variables with a larger number of possible discrete values make up a greater proportion of the Top-10 ranked variables.

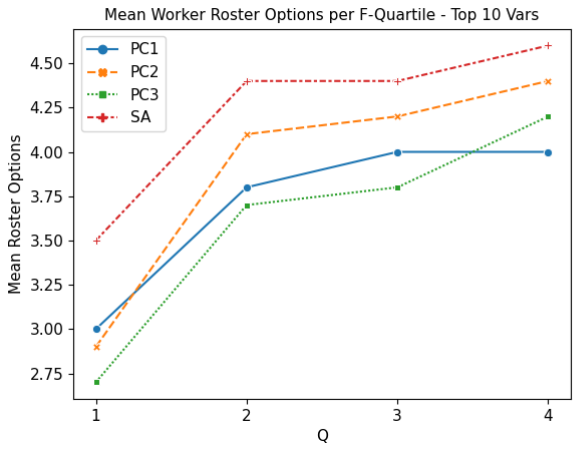

Figure 5: Mean Roster Options by F-Quartile - Top-10 Vars

The Figure shows that between the first and last quartile the mean number of options for variables increases from between 2.75 and 3.5 to between 4 and 4.6, indicating that over the course of the optimization, a larger proportion of the Top-10 variables by rank are being allocated to variables with a larger possible value pool. These are also typically the variables that take the longest to converge as measured by our variance calculations.

5 CONCLUSIONS

In this paper, we examine the extent to which the results of Sensitivity Analysis and Search Trajectory Analysis agree on search trajectories generated by a GA for a rostering problem. Both methods are capable of ranking workers based on their importance in roster assignments. SA results are derived from fitness sensitivity, while TA results come from PCA decomposition of search trajectories to show a workers contribution to the searches position in the created subspace. The main explanatory insights from our results are regarding the flexibility with which specific workers can be assigned to rosters and include where fitness value is most gained during the search. Our findings reveal that during the early generations of optimisation, there is a meaningful overlap in the top 10 ranked workers and overall worker rankings, although TA identifies a greater number of distinct workers. The overlap shows that the TA approach, while also identifying variables of high contribution, may also be identifying some level of variable-fitness sensitivity in the early stages of the search. The overlap decreases at the low added-value end of a search trajectory. As the TA approach explains approximately 46% of variance across the first three PCs, it is possible that a greater level of agreement between the two methods could be gained by using additional components. This however may come at the cost of the interpretability of the results. Further analysis is also planned to address the question of whether it is more accurate to measure the mean relevance of each variable across all components and at each generation, regardless of the trade-off between accuracy and interpretability.

We propose that TA, accounting for the algorithm's search behaviour, may offer additional insights into algorithmic behaviour such as those seen in our previous work. It should be noted that some overlap is expected among all explanatory methods. This overlap adds confidence that our approach is highlighting meaningful, explanatory insights to end users. By comparing TA to SA results, we highlight key groups of workers responsible for a large share of fitness. Furthermore, we provide end-users with a list of workers with flexible roster assignments and minimal fitness impact at the low added-value end of the search. This in turn helps to explain the GAs choices as it focuses on high-value individuals early in the search. Further explanation can be provided to the end-user through the explanation of the solutions themselves as we can show which workers are key to a good fitness solution and those who have additional assignment flexibility with lower fitness impact.

ACKNOWLEDGMENTS

This paper was written as part of a funded PhD project supported by The Data Lab and BT Group plc. This work has also greatly benefited from numerous scientific discussions at the Dagstuhl Seminar 22182 "Estimation-of-Distribution Algorithms: Theory and Applications".

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] Brownlee A.E.I. Cagnoni S. Iacca G. McCall J.A.W. Walker D. Bacardit, J. *"Evolutionary Computation and Explainable AI: a year in review"*. Late-breaking Abstracts, EvoStar Conference 2023, Brno, Czech Republic.

[5] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

[6] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[7] Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. *Advances in neural information processing systems*, 30, 2017.

[8] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

[9] 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.

[10] 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.

[11] Gabriela Ochoa, Katherine Malan, and Christian Blum. Search trajectories illuminated. *ACM SIGEVOlution*, 14:1–5, 07 2021.

[12] Gabriela Ochoa, Katherine Malan, and Christian Blum. Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics. *Applied Soft Computing*, 109:107492, 05 2021.

[13] Jonathan Wright, Mengchao Wang, Alexander Brownlee, and R.A. Buswell. Variable convergence in evolutionary optimization and its relationship to sensitivity analysis. 01 2012.

[14] Paulo Cortez and Mark J Embrechts. Opening black box data mining models using sensitivity analysis. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 341–348. IEEE, 2011.

[15] Paulo Cortez and Mark J Embrechts. Using sensitivity analysis and visualization techniques to open black box data mining models. *Information Sciences*, 225:1–17, 2013.

[16] 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.

[17] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[18] Samprit Chatterjee, AS Hadi, and B Price. Regression analysis by example john wiley & sons. *Inc., New York*, 2000.

[19] William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4):1–38, 2010.

[20] Krupesh Raikar. How to objectively compare two ranked lists in python, 01 2023.

[21] Alessia Sarica, Andrea Quattrone, and Aldo Quattrone. Introducing the rank-biased overlap as similarity measure for feature importance in explainable machine learning: A case study on parkinson's disease. In Mufti Mahmud, Jing He, Stefano Vassanelli, André van Zundert, and Ning Zhong, editors, *Brain Informatics*, pages 129–139, Cham, 2022. Springer International Publishing.

[22] Mary Dimitropoulaki, Mathias Kern, Gilbert Owusu, and Alistair McCormick. Workforce rostering via metaheuristics. In *Artificial Intelligence XXXV: 38th SGAI International Conference on Artificial Intelligence, AI 2018, Cambridge, UK, December 11–13, 2018, Proceedings 38*, pages 277–290. Springer, 2018.

[23] K. N. Reid, J. Li, A. E. I. Brownlee, M. Kern, N. Veerapen, J. Swan, and G. Owusu. A hybrid metaheuristic approach to a real world employee scheduling problem. In *Proc. of the Genetic and Evolutionary Computation COnference*. Prague, Czech Republic, 2019. Accepted, to appear.

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

[25] Kalyanmoy Deb, Karthik Sindhya, and Tatsuya Okabe. Self-adaptive simulated binary crossover for real-parameter optimization. In *Proceedings of the 9th annual conference on genetic and evolutionary computation*, pages 1187–1194, 2007.