Explaining a staff rostering problem by mining trajectory variance structures.

FYVIE, M., MCCALL, J.A.W., CHRISTIE, L.A., ZĂVOIANU, A.-C., BROWNLEE, A.E.I. and AINSLIE, R.

2023

Explaining A Staff Rostering Problem By Mining Trajectory Variance Structures

Martin Fyvie[0000−0001−8491−7008], John A.W. McCall[0000−0003−1738−7056], Lee A. Christie[0000−0001−8878−0344], Alexandru-Ciprian Zăvoianu[0000−0003−1003−7504], Alexander E.I. Brownlee[0000−0003−2892−5059], and Russell Ainslie[0000−0001−5398−1269]

[1] The Robert Gordon University, Garthdee Road, Aberdeen, UK
{m.fyvie, j.mccall, l.a.christie, c.zavoianu}@rgu.ac.uk
[2] University of Stirling, Stirling, UK
{alexander.brownlee}@stir.ac.uk
[3] The BT Group, Adastral Park, Ipswich, UK
{russell.ainslie}@bt.com

Abstract. The use of Artificial Intelligence-driven solutions in domains involving end-user interaction and cooperation has been continually growing. This has also lead to an increasing need to communicate crucial information to end-users about algorithm behaviour and the quality of solutions. In this paper, we apply our method of search trajectory mining through decomposition to the solutions created by a Genetic Algorithm — a non-deterministic, population-based metaheuristic. We complement this method with the use of One-Way ANOVA statistical testing to help identify explanatory features found in the search trajectories — subsets of the set of optimization variables having both high and low influence on the search behaviour of the GA and solution quality. This allows us to highlight these to an end-user to allow for greater flexibility in solution selection. We demonstrate the techniques on a real-world staff rostering problem and show how, together, they identify the personnel who are critical to the optimality of the rosters being created.

Keywords: Evolutionary Algorithms · PCA · Explainability · Population Diversity

1 Introduction

Artificial Intelligence (AI) including non-deterministic meta-heuristics such as Genetic Algorithms (GA) have seen a considerable increase in their application in domains involving end-user interaction and cooperation. These domains typically include Transport and Logistics [1] and Engineering [2]. An important aspect of this interaction is the need for some level of trust to be maintained between the end-users and the results generated. It is often recommended that such AI-powered systems follow a design philosophy that highly emphasises the interpretability of the results or the operations of these systems. This can include the development of methods capable of explaining these processes post-hoc, such as the recommendations of PHG [3]. More recently, this has become one of the core considerations of AI, as seen in its inclusion in the European Commission on Trustworthy AI publications [4]. Explainable AI (XAI) techniques have, in recent years, also seen an increase in attention, most likely driven by this growth in the adoption in these key domains.

This growth in XAI methods can be seen in XAI survey papers [5, 6] of these techniques and approaches which aim to help key stakeholders determine which AI system and XAI techniques are right for their application. The growing interest in the intersection of XAI with genetic and evolutionary algorithms can be seen in the newly created Genetic and Evolutionary Computation Conference (GECCO) XAI workshop in 2022 and follow-up in 2023.

There is a wide array of approaches to generating explanations regarding AI-generated solutions and their decision-making processes. These approaches include the extraction of some level of understanding from the sensitivities of the fitness function to specific variables. Sensitivity Analysis (SA), applications of which can be seen in [7, 8], is used to calculate solution fitness sensitivity to changes in variable values. Feature importance can also be mined from the fitness function through the use of Surrogate Modelling [9–11]. GAs, however, are often utilized as tools to enhance XAI techniques such in fuzzy logic systems [12] and counterfactual creation [13] and are rarely the focus of such analyses.

In this paper, we test our hypothesis that GA-generated search trajectories can be mined for features capable of adding a level of explanation to the resulting solutions. These explanations have the capacity to aid an end-user in understanding and interpreting the results and how they reflect the algorithm's search behaviour. This, in turn, may help build trust that the solutions provided are of high quality and relevant to the end users' goals. We then complement these results with those generated by an Analysis of Variance (ANOVA) technique to highlight any overlap between the two approaches.

The remainder of this paper is structured as follows: **Section 2** contains an overview of the experimental setup used to generate the search trajectories. This section includes our definition of a search trajectory, the optimisation problem definition and any algorithm-specific setup used. Shown in **Section 3** are the methods used to extract the explanatory features from the search trajectories. This section covers the implementation of Multiple Correspondence Analysis, Analysis of Variance implementation and the method by which we compare and merge the results - Weighted Ranked Biased Overlap. **Section 4** presents the results of our analysis and finally, **Section 5** contains our conclusions and plans for further work to extend this research.

2 Definitions and Target Algorithm

2.1 Search Trajectory Definition

During its run, a population-based algorithm visits a collection of solutions X. These are ordered by generation g and gathered into a search trajectory T as shown in Equation (1).

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

Here, N is the population size and n is the problem size. Thus, we define a *trajectory* as a list of $\mathbb{Z}^{gN}$ solutions of size n, drawn from the discreet integer space $\mathbb{Z}$ as outlined fully in Section 2.2. It is important to note that this definition of a trajectory is not limited to the integer domain. The approach has also be applied to problems in which the solutions lie in real-valued space $\mathbb{R}$ as shown in [14].

2.2 Problem Definition

The search trajectories we analyse in this paper were generated by solving a modified version of the staff rostering problem detailed in [15,16]. This problem aims to minimize the variance in the number of workers assigned to work on each day of the week, over a 3-month period. A pre-generated set of 100 roster patterns, varying in length from 2 to 13 weeks, details what days a worker will be required. Each of these rosters ensures two consecutive days off per week. Workers are assigned a sub-pool of between one and five potential rosters from the initial pool. All optimization runs are initialised with the same, pre-determined "starting state" which represents the initial configuration of rosters and the currently worked week of that roster. This allows for secondary goals aiming to minimize disruption to the workforce. In this problem, all workers may change from their currently assigned week to a different week within their initial roster however only a fraction of the workforce are allowed to change to a new roster in their sub-pool.

Table 1: Solution Representation

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

(a) Sol. Extract

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

(b) x_2 Roster-Week

An example solution representation can be seen in Tables 1a and 1b. Here, each worker is represented by the variable $x_i, 1 \leq i \leq n$ in Table 1a. The value each variable can take refers to the index of a variable-specific table containing all possible combinations of their roster sub-pool and starting weeks. Table 1b shows an example of this index table for variable x_2. The solution shown indicates that variable x_2 – with a value of 33 – represents Roster 7, starting week 3. This in turn would be understood by the user as Worker x_2 beginning the 3-month period using the working hours determined by that selection and would repeat roster 7 from week 1 if the roster is completed before the end of the three-month period.

We use an "attendance matrix" A_{kd} which, in our 3-month problem, is a 12×7 matrix of the sum of the workers assigned to work on week k, day d. These totals are determined by the variable values in solution X which will in turn determine the total number of workers scheduled for each day d, in all 12 weeks of k. This maps the original problem definition to our trajectory definition.

The range between total workers assigned for each day is calculated by taking the minimum and maximum of matrix $\mathbf{A}$ for each column d, to calculate the normalized range of column d as shown in Equation (2). This in turn us used to calculate the fitness value of a solution, shown in Equation (3) subject to the constraint detailed in Equation (4).

$$R_d = \frac{\underset{d}{max}\,(a_{kd}) - \underset{d}{min}\,(a_{kd})}{\underset{d}{max}\,(a_{kd})} \tag{2}$$

$$\text{minimize:} \sum_{1 \le d \le 7} w_d R_d^2 + \left(\sum_{n \in x} P_n \right) S \tag{3}$$

$$\text{subject to:} \sum_{i=1}^{n} \mathrm{CV}_i \le 0.2 \cdot \mathrm{len}(x) \tag{4}$$

The cost function shown in Equation (3) aims to minimise the overall range between the number of workers assigned to work on each of the week days. This calculation has a set of weights, w, applied to each day of the week to reduce the impact of the lower availability of workers during the weekend. These are applied to each day d, with the values being set at $w = (1, 1, 1, 1, 1, 10, 10)$. This was done as "...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" [15]. The function contains constraints designed to minimize the disruption to the workforce. The hard constraint, shown in Equation (4), defines CV as the total number of workers who have been assigned to a new roster from their sub-pool. This constraint aims to limit the total number of workers from being assigned new rosters to 20% of the total workforce. The cost function contains a soft constraint linked to the second summand of x, the set of all variables in a solution, and $P = (p_n)$, a binary array in which $p_n = 1$ if the value of variable n results in two consecutive Saturdays being scheduled or 0 otherwise, due to changing from the initial Roster pattern and week to those outlined in a solution. This soft constraint adds a small penalty of $S = 0.01$ for each violation of this constraint in each solution to help reduce the total occurrences of this. As the aim is to reduce the total range to 0, a minimum value of 0 would be achieved should all ranges be reduced to 0 and no consecutive Saturdays be worked. The source data files used for rosters and allocations can be found here [17].

2.3 Algorithm Runs

The trajectories representing runs of a GA with a population μ form the target of the explanation techniques presented in this paper. Here, μ is the starting population of solutions and the resulting next generation of solutions is created through the application of the internal operators of the GA. The operators are Selection, Crossover and Mutation which are applied to the parent population to generate the child population of solutions. Shown in Table 2 are the run settings used to create the datasets for this paper.

Table 2: Algorithm Run Settings

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

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

The Python Multi-Objective Optimisation (PYMOO) library [18] was used to implement the GA with the required parameters. After some initial testing, the values in Table 2 were selected to allow for reasonable solution convergence however it is important to note that refining the algorithm's performance was not a consideration in this study. Provided that higher quality solutions were being generated, we are able to continue with our analysis. As this was the case, where possible the default values and operators outlined in the PYMOO documentation were used.

As the mutation used was a polynomial mutation [19] function, details of which can be found in [20], the setting **eta** was set to 3. The higher the value is set the more similar and less mutated the child solution will be. The solutions were encoded as discrete variable strings, in which each value in the string represented the value given to a specific worker. These values represented the index of a worker-specific table that contained all possible combinations of Rota and Starting Weeks. This representation required an implementation of the GA that could account for possible disruptions to a solution introduced by the internal operators.

3 Feature Extraction

3.1 Multiple Correspondence Analysis

The method to extract explanatory features outlined in this paper utilizes the process of trajectory decomposition into a set of derived variables or dimensions. There are many decomposition techniques available, and previous work has included the use of Principal Component Analysis [21] to that end. In this paper, we apply a variation of correspondence analysis called Multiple Correspondence Analysis (MCA) to allow for the decomposition of the dataset in which the variables, while taking integer values, are in practice nominal which would reduce the applicability of PCA directly. As shown in [22], it is possible to link MCA to PCA such that the application of an un-standardized PCA to an indicator matrix such as a Transformed Complete Disjunctive Table (TCDT), can lead to the same results as MCA. This process involves the creation of the Complete Disjunctive Table (CDT) by replacing the categorical variables with one-hot encoded dummy variables. This must be transformed as seen in Equation (5) in which the value of the CDT table, x_ik is transformed using y_{ik}, the proportion of solutions containing that value.

$$x_{ik} = y_{ik}/P_k - 1 \tag{5}$$

The application of PCA to the resulting TCDT provides us with the necessary directional vectors from our dataset as both approaches project the values to a lower-dimensional Euclidean space for the purpose of data analysis. Shown in Equation (6), this creates a set of m, $n \times 1$ orthonormal eigenvectors in $\mathbb{R}^n$. The elements of the p^i vectors represent the weighting of each variable, $[p^i_1, \ldots, p^i_n]$. These coefficients help describe the contribution of each variable to the corresponding principal component in terms of maximizing the variance in the dataset through a best-fit hyperplane.

$$\begin{aligned} P &= [p^1, \ldots, p^m], m \leq n \\ p^i &= [p^i_1, \ldots, p^i_n] \end{aligned} \tag{6}$$

With these, we can calculate the Mean Squared Cosine (MSC) value associated with each variable in the problem. Equations (7) to (9) outline this process.

$$PC_{x_n c} = \sqrt{\lambda_m} \cdot \text{Factor Loadings}(x_n, p^m) \tag{7}$$

$$PC_{x_n} = \sqrt{\sum_{c=1}^{num_c} \lambda_m \cdot \text{Factor Loadings}(x_n, p^m)} \tag{8}$$

$$\text{MSC}(x_n) = \frac{1}{num_c} \sum_{c=1}^{num_c} \left(\frac{PC_{x_n c}}{PC_{x_n}} \right)^2 \tag{9}$$

The principal coordinate of variable x_n in category c is denoted as $PC_{x_n c}$, while PC_{x_n} represents the same variable's principal coordinate across all categories, whose count is num_c. The eigenvalue of component p^m is λ_m, and the associated loadings are the Factor Loadings(x_n, m). The Mean Squared Cosine (MSC) value, a measure of the proportion of variance captured by each category in each variable, is the squared cosine of the angle between a variable and its categories in the Multiple Correspondence Analysis (MCA) space. A higher MSC value indicates a stronger relationship between the categories and the variable in the MCA space, implying greater importance of those categories in capturing the structure and variability in the MCA analysis. We decompose the search trajectories from each optimization run, resulting in subspaces each characterizing the algorithm's search trajectory in terms of the variance of one variable and its categories. This reveals which variables are crucial to the algorithm's position on the fitness gradient. The output of this process is a collection of datasets representing each variable's influence, ranked in ascending order from 1 to n (141), where 1 is the least influential and n is the most, across a subset of the m eigenvectors created.

3.2 Analysis of Variance

To gain a better understanding of the trajectory analysis results, we employ Analysis of Variance (ANOVA), which is a statistical method for the comparison of means across multiple groups. This is used to generate a comparative set of variable rankings. In our datasets, we can use ANOVA to compare the means of each variable in our solutions to the dependent variable - solution fitness. This analysis technique is used to detect whether there is a relationship between each variable and the fitness of a solution. This is done by using the sum of squares between value groups and the sum of squares within groups. The resulting *p-value* can be used to indicate whether any detected relationship is statistically significant. For the purpose of this paper, we consider all variables to be independent. This decision means that we can apply the ANOVA test to each variable-fitness pair separately and calculate the "partial eta squared" value for each pair. Partial eta squared is a measure of effect size in ANOVA that represents the proportion of total variance that is explained by an effect while controlling for other effects. To calculate the partial eta squared values, we use the Python "statsmodel" package [23] implementation of ANOVA. Equations (10) to (12) show how we calculate the partial eta value (η_p^2). Here, k is the number of solutions in our trajectory (gN), n is the number of variables, x_{ij} is the jth variable in the solution i of the whole trajectory. The mean value of all variables in solution i is shown as $\bar{x}_i$.

$$SS_{\text{within}} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i)^2 \tag{10}$$

$$SS_{\text{between}} = \sum_{i=1}^{k} n_i (\bar{x}_i - \bar{x})^2 \tag{11}$$

$$\eta_p^2 = \frac{SS_{\text{between}}}{SS_{\text{between}} + SS_{\text{within}}} \tag{12}$$

We use the value $\bar{x}_i$, in conjunction with the resulting *p-value* generated for each variable, to determine the influence that the variable x_i has on Fitness and whether that influence is statistically significant. For this paper, if $p < 0.05$, we reject the null hypothesis that x_i has no measurable influence on fitness across all trajectories. Once complete, we use the partial eta to rank each variable in terms of the size of their effect on the fitness measured. It is important to note that as all 100 optimization runs were performed separately, the ANOVA analysis must be performed a total of 100 times, each resulting in a set of partial eta values and $p-values$. There are several methods of accommodating this approach that would allow us to use the $p-values$ from across multiple runs to determine the significance of our findings.

One process is Bonferroni [24] however, as we are using 100 total runs, this would require us to adjust to *p-value* threshold to a prohibitively small value due to how the Bonferroni correction is calculated. We opted to use the less conservative Benjamini-Hochberg [25] process to instead account for the false discovery rate (FDR) that using 100 runs may introduce. This method, as shown in Algorithm 1 involves the ranking of all calculated $p-values$ before calculating the "Benjamini-Hochberg critical value".

Algorithm 1 Benjamini-Hochberg Procedure

```
Require: p-values $P[1..m]$, false discovery rate $Q$
Ensure: List of rejected hypotheses $R$
 1: Arrange the p-values in ascending order: $P[1] \leq P[2] \leq ... \leq P[m]$
 2: Initialize an empty list of rejected hypotheses: $R = []$
 3: for $i = m$ to 1 do
 4:     Calculate the Benjamini-Hochberg critical value: $BH = \frac{i}{m}Q$
 5:     if $P[i] \leq BH$ then
 6:         Add $i$ to the list of rejected hypotheses: $R = R + [i]$
 7:         Break the loop
 8:     end if
 9: end for
10: return $R$
```

This is then used to determine, for each variable, what the relevant adjusted *p-value* should be to keep the FDR below 0.05 and results that are higher are rejected and removed from the dataset.

3.3 Weighted Ranked Biased Overlap

To facilitate the comparison of the variable rankings produced by both MCA and ANOVA, we use a method known as Weighted Rank Biased Overlap (WRBO) [26], which has been used in the past to increase the interpretability of machine learning results [27]. This method allows the comparison of ranked-lists with the added benefit that both lists can be of varying length and do not need to contain all of the same elements. It is this ability that led us to use WRBO over more classical rank-comparison methods such as Spearman Rank Correlation or the Kendall Tau method. A further benefit of this method is that it can place a higher weighing to elements at the top of a list which can be customised. The output of this method is a similarity score representing the proximity between both lists. This score takes a value of $[0, 1]$, with 1 showing a complete overlap and 0 no similarity between the order of the list elements and the number of shared elements. This process also takes into account any weightings given to the top elements. This process can be seen in Algorithm 2 which was created from a Python implementation of the WRBO function outlined in [28].

Algorithm 2 Rank Biased Overlap (RBO) Python

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

As the results of our analysis are sets of ranked variables, from most to least influential, the ability of WRBO to set a higher weight in its similarity calculation to the top members of a list was of great benefit. The method has a parameter that can increase the weighting of a top subset of variables. This WP value is dependent on the presumed size of the lists. For our purposes, setting this value at 0.9 results in the top 10 variables being responsible for 85.56% to the total scoring. This is done to allow for the most influential variables found in both methods to influence the scoring more than the other, lower influence variables. It is also possible to set the WP to 0.98. This will result in requiring the first 100 items in the list to result in a similar weighting of $\sim 85\%$.

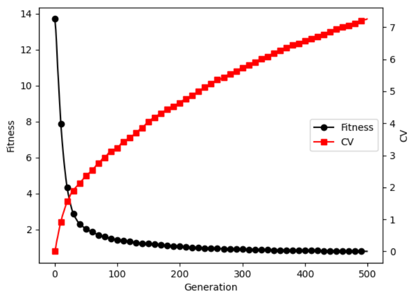
With this rank-comparison method, we are able to generate a measure of similarity between the findings of the MCA and ANOVA analyses. As ANOVA measures the impact that varying a variable has on the variance in fitness, we can use this measure to show any overlap. This overlap may represent some level of shared findings, highlighting that our trajectory mining method may also be able to detect some level of structure that the ANOVA approach is discovering.

4 Results

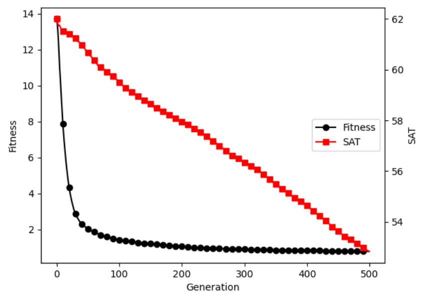
In this section, we analyse and interpret the results of the optimization of the rostering problem and the features we are able to mine from the search.

4.1 Rostering Results

The results of running the optimization a total of 100 times can be seen in Figures 1a and 1b. These show the mean results for fitness and usage of both the hard and soft constraints in the problem. Figure 1a shows the results between solution fitness and the number of variables that were assigned a value resulting in a change of rosters (CV). We see that averaging over 100 runs, the GA was able to find considerably better solutions than the initial starting state of the problem – from a mean fitness of 14 to approximately 0.7. Within the first 100 generations, the value of CV increases from 0 to 6.



(a) CV vs Fitness



(b) SAT vs Fitness

Fig. 1: Constraints vs. Mean Fitness

Over the course of the remaining 400 generations, this value continues to increase but at a lower rate. This shows that within the first 100 generations, a significant improvement in solution quality is achieved with approximately 6 roster reassignments. As the rate of fitness change slows, we see that the GA makes small, incremental improvements to solutions at the expense of adding new roster assignments. Figure 1b shows the results of fitness and the soft constraint SAT - the number of consecutive Saturdays assigned. The results show a slow, steady reduction in this value over all 500 generations from an initial value of 62 to approximately 52.

The changes in daily range values can be seen in Figure 2. Here, we show the distribution of range values for each day of the week over all 100 runs. Figure 2 shows the range values at 3 different generations - 5, 100 and 500. Between generation 5 and 100 we see a clear reduction in the mean range values across all runs for all days of the week except Saturday, with this day showing a small increase from 0.075 to 0.078. We also see a reduction in the upper limit of ranges seen on Mon, Tue and Wed. Between generations 100 and 500 we see an increase in the mean range on Mon and Sat while the other days show either a reduction or little change. As the fitness value continues to reduce over this period, solutions with a higher range value on some days are being found that achieve a higher quality solution with a more balanced overall range across the week.

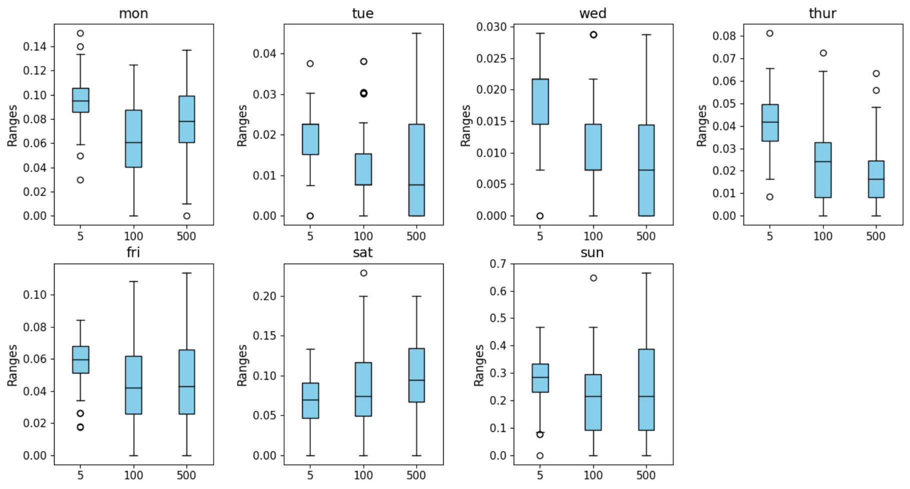
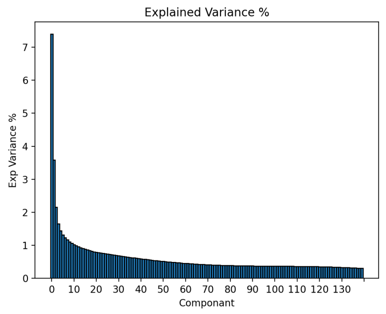


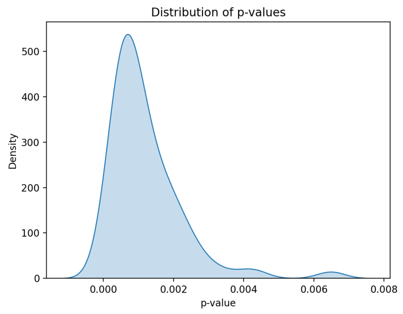
Fig. 2: Range Results - Unweighted

4.2 MCA and ANOVA Feature Results

After the results of the ANOVA testing were gathered, we performed the Benjamini-Hochberg *p-value* adjustment method to allow for the comparison of all 100 runs. Shown in Figures 3a and 3b is the explained variance by component for MCA and the distribution of $p-values$ resulting from the adjustment respectivly. The ANOVA results show that all included values in the analysis are below the set threshold of 0.05, with nearly all being below 0.003.



(a) MCA Explained Variance



(b) ANOVA P-Value Distribution

Fig. 3: MCA and ANOVA Results

This process resulted in roughly 30% of the variable partial eta values being removed from the dataset as the associated *p-value* for those results did not meet the required threshold of 0.05. This ensured that the variable rankings produced via the ranking of the ANOVA partial eta values remained viable.

Similarly, the results of the MCA decomposition were inspected to ensure that the results were representative of the level of variance captured by this method. Seen in Figure 3a is the percentage of variance explained by each of the components generated when averaged across all 100 runs. These results show that the first component explains a mean value of around 7.4% of the variance in the dataset. The level of explained variance from component 2 onwards drops off significantly. As the first component explains a relatively small amount of variation, we show the results of using multiple subsets of the components for comparison.

Table 3 shows the WRBO similarity scores when comparing both the Top-10 ranked variables and all 141 variables between the MCA and ANOVA methods. We show the similarity between variable ranks using all components, the first 50, 10, 5 and 1st component. We also show the effect of altering the WRBO WP value from 0.0 to 0.98, shifting the weighting from 85% attributed to the Top-10 to 100 for the same effect. The highest similarity score for each test is shown in bold.

Table 3: WRBO Similarity Scores - MCA to ANOVA

	WP = 0.9		**WP=0.98**	
Dataset	**WRBO**	**WRBO-Top 10**	**WRBO**	**WRBO-Top 10**
All Comp	0.234	0.166	0.495	0.194
50 Comp	0.301	0.232	0.524	0.293
10 Comp	0.363	0.322	0.568	0.311
5 Comp	0.449	0.394	0.607	0.327
1 Comp	**0.664**	**0.606**	**0.694**	**0.449**

From these results, we can see that the lowest level of similarity between both methods across all tests comes from using all components generated by MCA. The highest similarity scores are found when using only component 1, the highest explained variance component. As more are added, the overall similarity score reduces. This holds true for both $WP = 0.9$ and $WP = 0.98$. The highest level of similarity is found when $WP = 0.98$ and only 1 component is used, giving a score of 0.694, showing the considerable overlap in findings between the two methods. This overlap in findings can be seen in Tables 4 and 5 in which we show the variables identified as being both high and low importance. Table 4 shows the variables identified as most influential for both MCA and ANOVA. Also shown are the results for the top 1, 5, 10, 50 and all components for comparison. Highlighted in bold and brackets are any variables found in both the ANOVA and any components' Top-10 list. Here, we show that variables of both high MCA-cosine squared ranking and high ANOVA partial-eta ranking in common are [121, 65, 1 and 60]. As these are ranked highly by both methods, these would be considered highly influential in both algorithm search direction and impact on fitness, based on the MCA and ANOVA results respectively. The results in this table also reflect the higher WRBO similarity scores, as the 1 component results show 4 overlapping variables. The remainder shows only 3 except when using all components, where the overlap drops to one, mirroring the decreasing WRBO scores. This would indicate that when viewing these results from the perspective of an end-user, workers [121,65,1,60] would be of particularly high interest due to their high influence. Further explanations may be gained from a closer assessment of their working pattern preferences as those assigned to these workers are consistently required for high-quality solutions to the scheduling problem.

Table 4: Most Influential Variables by Dataset

	ANOVA	1 Comp	5 Comp	10 Comp	50 Comp	All Comp
Rank	Var	Var	Var	Var	Var	Var
141	**(121)**	**(121)**	**(1)**	96	**(1)**	**(1)**
140	**(65)**	**(65)**	**(65)**	**(1)**	96	70
139	51	**(1)**	**(121)**	**(65)**	**(65)**	135
138	**(1)**	96	96	**(121)**	135	67
137	33	135	135	135	48	76
136	**(60)**	**(60)**	128	119	91	**65**
135	129	72	72	128	67	68
134	126	128	119	67	119	96
133	110	48	48	72	72	91
132	66	119	68	48	**(121)**	72

The results in Table 5 show the overlap in rankings between the MCA and ANOVA methods for the lowest-ranking variables. These variables would be considered to have a low impact on fitness due to their low partial eta value, and of low influence on the overall search path due to their low cosine squared ranking. The overlap between the two methods identifies variables [21, 137 and 69]. These results also show a similar pattern to the highly ranked variable such that, as more components are used in the calculation, the lower the overlap and similarity between ANOVA and MCA becomes. To an end-user, these results could help highlight that workers [21, 137 and 69] have a lower impact on solution quality and algorithm direction. The results would suggest that these workers have a higher capacity for roster allocation with minimal impact on the overall quality of the schedule, allowing for more customization to accommodate any additional goals.

Table 5: Least Influential Variables by Dataset

	ANOVA	1 Comp	5 Comp	10 Comp	50 Comp	All Comp
Rank	Var	Var	Var	Var	Var	Var
10	118	97	97	97	97	97
9	**(21)**	15	15	4	4	15
8	88	**(21)**	4	15	15	4
7	**(137)**	4	**(21)**	**(21)**	**(21)**	**(21)**
6	38	77	9	9	9	77
5	25	9	77	77	77	9
4	83	**(69)**	93	93	**(69)**	36
3	84	79	79	79	93	93
2	**(69)**	93	**(69)**	**(69)**	79	105
1	37	**(137)**	105	105	105	**(69)**

5 Conclusions and Future Work

In this paper, we used Multiple Correspondence Analysis to analyze the search trajectories generated by a Genetic Algorithm for a staff rostering problem. We calculated the squared cosine value for each variable from the subspace, which was derived from decomposing the trajectories. These results were then compared with the outcomes of one-way ANOVA testing on the same datasets, leading to a set of statistically significant measurements of the partial-eta value. This value measures the relative impact a variable has on fitness variance in the data. We ranked both sets of results and used the Weighted Rank-Biased Overlap similarity metric to measure the overlap in findings. Our results show a significant overlap (0.69) when only the first component is used, averaged across all 100 runs. However, the addition of more components showed diminishing returns, possibly because the first component captures key structures in a widely spread dataset. The significant level of residual variance might contain a lot of noise, which could disrupt the process as more components are added. Further study would be needed to identify any missed structure in the residual variance.

Our experiments identified key variable subsets, corresponding to individual workers, using both methods together. The overlap between the two methods in the top and bottom rankings provides a subset of variables that have either a high or low influence on the search path and fitness impact. This gives end-users a way to identify key individuals and those who, due to their lower impact, could be moved to another observed rota schedule with minimal disruption. This tool is valuable to end-users, helping to explain key drivers towards high-quality solutions and the capacity for minimal impact change.

References

1. Abduljabbar, R., Dia, H., Liyanage, S., Bagloee, S.A.: Applications of Artificial Intelligence in Transport: An Overview. Sustainability 11(1), 189 (2019)

2. Brownlee, A.E., Wright, J.A., He, M., Lee, T., McMenemy, P.: A Novel Encoding for Separable Large-scale Multi-objective Problems and its Application to the Optimisation of Housing Stock Improvements. Applied Soft Computing 96, 106650 (2020)
3. Hall, A., Ordish, J., Mitchell, C., Richardson (nee Murfet), H.: Black box medicine and transparency - interpretability by design framework. Tech. Rep. MSR-TR-2020-53, PHG Foundation (February 2020), https://www.microsoft.com/en-us/research/publication/black-box-medicine-and-transparency-interpretability-by-design-framework/
4. European Commission: Ethics Guidelines for Trustworthy AI. https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai (2021)
5. Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F.: Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI. Information Fusion 58, 82–115 (2020), https://www.sciencedirect.com/science/article/pii/S1566253519308103
6. Dwivedi, R., Dave, D., Naik, H., Singhal, S., Omer, R., Patel, P., Qian, B., Wen, Z., Shah, T., Morgan, G., Ranjan, R.: Explainable AI (XAI): Core Ideas, Techniques, and Solutions. ACM Comput. Surv. 55(9) (2023), 10.1145/3561048
7. Wright, J., Wang, M., Brownlee, A., Buswell, R.: Variable Convergence in Evolutionary Optimization and its Relationship to Sensitivity Analysis. Building Simulation and Optimization 2012, Loughborough, UK pp. 102–109 (2012), https://repository.lboro.ac.uk/articles/conference_contribution/Variable_convergence_in_evolutionary_optimization_and_its_relationship_to_sensitivity_analysis/9438080
8. Cortez, P., Embrechts, M.J.: Using Sensitivity Analysis and Visualization Techniques to Open Black Box Data Mining Models. Information Sciences 225, 1–17 (2013)
9. Jin, Y.: Surrogate-Assisted Evolutionary Computation: Recent Advances and Future Challenges. Swarm and Evolutionary Computation 1(2), 61–70 (2011)
10. Wallace, A., Brownlee, A.E.I., Cairns, D.: Towards Explaining Metaheuristic: Solution Quality by Data Mining Surrogate Fitness Models for Importance of Variables. In: Bramer, M., Ellis, R. (eds.) Artificial Intelligence XXXVIII. pp. 58–72. Springer International Publishing, Cham (2021)
11. Singh, M., Brownlee, A.E.I., Cairns, D.: Towards Explainable Metaheuristic: Mining Surrogate Fitness Models for Importance of Variables. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. p. 1785–1793. GECCO '22, Association for Computing Machinery, New York, NY, USA (2022)
12. Duygu Arbatli, A., Levent Akin, H.: Rule Extraction from Trained Neural Networks using Genetic Algorithms. Nonlinear Analysis: Theory, Methods & Applications 30(3), 1639–1648 (1997)
13. Sharma, S., Henderson, J., Ghosh, J.: Certifai: Counterfactual Explanations for Robustness, Transparency, Interpretability, and Fairness of Artificial Intelligence Models. Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (2020)
14. Fyvie, M., McCall, J.A.W., Christie, L.A., Brownlee, A.E.: 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 (2023)

15. Dimitropoulaki, M., Kern, M., Owusu, G., McCormick, A.: Workforce Rostering via Metaheuristics. In: Bramer, M., Petridis, M. (eds.) Artificial Intelligence XXXV. pp. 277–290. Springer International Publishing (2018)
16. Reid, K.N., Li, J., Brownlee, A., Kern, M., Veerapen, N., Swan, J., Owusu, G.: A Hybrid Metaheuristic Approach to a Real World Employee Scheduling Problem. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 1311–1318. GECCO '19, Association for Computing Machinery, New York, NY, USA (2019)
17. Fyvie, M., McCall, J.A., Christie, L.A., Zăvoianu, A.C., Brownlee, A.E., Ainslie, R.: Explaining A Staff Rostering Problem By Mining Trajectory Variance Structures Definition (2023), https://github.com/rgu-subsea/mfyvie_sgai2023_varstruct.git
18. Blank, J., Deb, K.: Pymoo: Multi-Objective Optimization in Python. IEEE Access 8, 89497–89509 (2020)
19. Deb, K., Deb, D.: Analysing Mutation Schemes for Real-Parameter Genetic Algorithms. International Journal of Artificial Intelligence and Soft Computing 4, 1–28 (2014)
20. Deb, K., Sindhya, K., Okabe, T.: Self-Adaptive Simulated Binary Crossover for Real-Parameter Optimization. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation. p. 1187–1194. GECCO '07, Association for Computing Machinery, New York, NY, USA (2007)
21. Fyvie, M., McCall, J.A., Christie, L.A.: Towards Explainable Metaheuristics: PCA for Trajectory Mining in Evolutionary Algorithms. In: Bramer, M., Ellis, R. (eds.) Artificial Intelligence XXXVIII. pp. 89–102. Springer, Springer International Publishing (2021)
22. Pagès, J.: Multiple Factor Analysis by Example Using R. Chapman and Hall/CRC, 1 edn. (2014)
23. Seabold, S., Perktold, J.: Statsmodels: Econometric and statistical modeling with python. Proceedings of the 9th Python in Science Conference 2010 (01 2010)
24. Dunn, O.J.: Multiple Comparisons Among Means. Journal of the American Statistical Association 56(293), 52–64 (1961)
25. Benjamini, Y., Hochberg, Y.: Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. Journal of the Royal Statistical Society: Series B (Methodological) 57(1), 289–300 (1995)
26. Webber, W., Moffat, A., Zobel, J.: A Similarity Measure for Indefinite Rankings. ACM Transactions on Information Systems (TOIS) 28(4), 1–38 (2010)
27. Sarica, A., Quattrone, A., Quattrone, A.: Introducing the Rank-Biased Overlap as Similarity Measure for Feature Importance in Explainable Machine Learning: A Case Study on Parkinson's Disease. In: Brain Informatics: 15th International Conference, BI 2022, Padua, Italy, July 15–17, 2022, Proceedings. p. 129–139. Springer-Verlag, Berlin, Heidelberg (2022)
28. Raikar, K.: How to Objectively Compare Two Ranked Lists in Python (2023), https://towardsdatascience.com/how-to-objectively-compare-two-ranked-lists-in-python-b3d74e236f6a