





# Towards Explainable Metaheuristics: PCA for Trajectory Mining in Evolutionary Algorithms

Martin Fyvie<sup>(✉)</sup> , John A. W. McCall<sup></sup>, and Lee A. Christie<sup></sup>

The Robert Gordon University, Garthdee Road, Aberdeen, UK  
{m.fyvie,j.mccall,l.a.christie}@rgu.ac.uk

**Abstract.** The generation of explanations regarding decisions made by population-based meta-heuristics is often a difficult task due to the nature of the mechanisms employed by these approaches. With the increase in use of these methods for optimisation in industries that require end-user confirmation, the need for explanations has also grown. We present a novel approach to the extraction of features capable of supporting an explanation through the use of trajectory mining - extracting key features from the populations of NDAs. We apply Principal Components Analysis techniques to identify new methods of population diversity tracking post-runtime after projection into a lower dimensional space. These methods are applied to a set of benchmark problems solved by a Genetic Algorithm and a Univariate Estimation of Distribution Algorithm. We show that the new sub-space derived metrics can capture key learning steps in the algorithm run and how solution variable patterns that explain the fitness function may be captured in the principal component coefficients.

**Keywords:** Evolutionary algorithms · PCA · Explainability · Population diversity

## 1 Introduction

Non-Deterministic Algorithms such as population-based meta-heuristics have seen an increase in use in applications that involve end-user interactions such as transport route planning, delivery scheduling and medical applications. This increase has highlighted the need for the decision processes behind these system to be more understandable by end-users. This in turn may help build a level of trust in the solutions generated by these systems, as seen in conclusions and recommendations of the Public Health Genetics (PHG) foundation [1].

Two significant metaheuristic approaches are Genetic Algorithms (GA) and Estimation of Distribution (EDA) algorithms. Both are evolutionary algorithms and explore a solution space using a population-based search metaheuristic. As a GA explores the search space, solution populations generated represent the

implicitly learned structure of the problem it is solving. The EDA similarly represents this but also generates a sequence of explicit probabilistic models of the problem structure. Problem Structure refers to the graphical dependency relationship between solution variables and their joint influence on fitness value. This has been variously interpreted in the EDA literature through Bayesian, Markov or Gaussian probabilistic models [2]. For both GAs and EDAs, the collected populations generated over the course of a run can be considered the trajectory through the search spaces that the algorithm has taken as it converges on an ideal or near-ideal solution. These trajectories reflect the implicit knowledge gained.

We hypothesize that the trajectories generated in this process can be mined for valuable information regarding population changes that can aid in generating explanations for end-users. Our approach involves the projection of the high-dimension solutions space to a lower dimension space that can be used to generate more easily understood visualisations and provide a possible source of new metrics. This is accomplished through the application of Principal Components Analysis (PCA). The results can then be used to generate explanations with the aim of increasing an end-user understanding of the problem being solved and the process by which the algorithms have arrived at the provided set of solutions.

Previous work covering the visualisation of algorithm trajectories using PCA can be seen in [3] and more recent methods in which local optima networks are used to generate search trajectory networks for different algorithm runs in [4]. Other examples of work involving the exploration of algorithm paths via dimension reduction include [5] in which Sammon mapping is explored as a method of reduction for visualisation and [6] in which Euclidean Embedding is applied. These works focus on the visualisation of an algorithm through the search space however the approach taken in this paper using PCA has the potential to be used as a method extracting features from algorithm paths. These features can then be used to help support explanations by highlighting learning steps in the algorithm run and solution variable patterns that describe the fitness function.

The rest of the paper is structured as follows. Section 2 outlines the experimental setup that covers the algorithms used and the problems they were used to solve. Section 3 outlines the concept of Entropic Divergence and how this is used as measure of population diversity change. This is followed by the background method used to translate the algorithm trajectories into a lower dimension space as well as the new metrics derived from that space for comparison to the Entropic Divergence measurement. Section 4 highlights the results and performance between the newly created population metrics and the Entropic Divergence are shown and discussed. This section also highlights the findings regarding problem structure and post-PCA projection variable loading's. Section 5 sets out our conclusions based on the results of these tests.

## 2 Experimental Setup

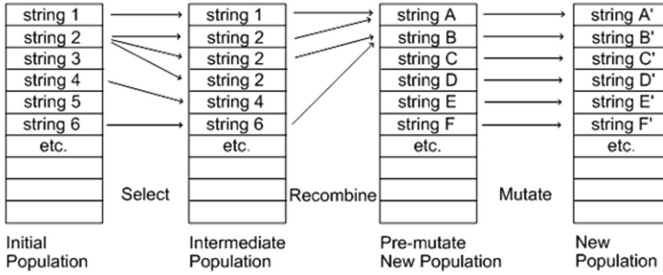
### 2.1 Algorithm Runs

Two population-based solvers were selected to generate a series of population trajectories for use in this study. These were a Genetic Algorithm (GA) and a modified Population Based Incremental Learning (PBIL) Algorithm [7, 8] in which a negative mutation rate and mutation shift value is introduced. These algorithms were selected for the purpose of comparing the results of a univariate solver and a more traditional genetic algorithm on problems with different structure. Each algorithm was run on the set of outlined problems in order to generate the trajectories used in the analysis phase of this trial.

**The Genetic Algorithm** used was an adaptation of the Canonical Genetic Algorithm (CGA) [9]. Figure 1 shows the main steps involved as the GA generates new populations during an optimisation run.

**Table 1.** GA run specifications

P	n	runs	maxGen	mutRate	Selection	Crossover
100	40	100	100	0.005	Tournament	Random 1-Point



**Fig. 1.** CGA stages taken from [10]

Table 1 outlines the values used in the running of the GA during these experiments.

**P** shows the number of solutions in each population. **maxGen** is the maximum number of generations before termination. **mutRate** is the mutation rate within the GA. **Selection** is the selection method used within the GA for solution comparison and reproduction. **Crossover** is the crossover type used in this trial with a rate of 1 and so occurs each generation. **n** is the problem length. **runs** is the number of runs for each problem the GA ran for.

**Population Based Incremental Learning (PBIL)** is a form of Estimation of Distribution algorithm. The probability vector is updated and mutated each generation as seen in Eqs. 1 and 2:

**Table 2.** PBIL run specifications

P	n	runs	maxGen	mutRate	mutShift	learnRate	nlearnRate
100	40	100	100	0.005	0.05	0.1	0.075

$$p(X_1, \dots, X_N) = \prod_{i=1}^N p(X_i) \quad (1) \quad p(X_i) = \frac{1}{N} \sum_{j=1}^N x_{ij} \quad (2)$$

Here the vector of marginal probabilities  $P_V = (p(X_1), \dots, p(X_n))$  is created by calculating the arithmetic mean of each variable  $X$  in a population of size  $N$ . As the solutions are comprised of bit strings we will see that values will range from 0 to 1. Table 2 outlines the values used for the PBIL algorithm in this trial. Additional to these, the PBIL used a **mutShift** value that was applied to mutated probability vector values. **learnRate** is the learning rate of the algorithm as is the **nlearnRate** which shows the negative learning rate penalty if the best solution matches the worst in a solution when updating the probability vector.

## 2.2 Benchmark Problems

**The 1D Checkerboard** function scores the chromosome based on the sum of adjacent variables that do not share the same value [11]. The function is seen here in Eq. 3.

$$CHECK_{1D}^l(x) = \sum_{i=0}^{l-2} \left\{ \begin{array}{l} 1, x_i \neq x_{i+1} \\ 0, x_i = x_{i+1} \end{array} \right\} \quad (3)$$

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

**The Royal Road** function scores chromosomes based on collections of variable values based on a specified set of schema that the solution must fulfil in order to score an optimal value [12]. below, Eq. 4 specifies the fitness function for the royal road problem with a schema block size of 5, as used in this experiment.

$$R_1(x) = \sum_{i=1}^5 \delta_i(x) o(s_i), \text{ where } \delta_i(x) = \left\{ \begin{array}{l} 1 \text{ if } x \in s_i \\ 0 \text{ if otherwise} \end{array} \right\} \quad (4)$$

As noted in [12] the equation represents the fitness function, such that  $R_1$  is a sum of terms relating to partially specified schema. The schemata are subsets of solutions that match the partial specification,  $s_i$ . As an example, one partially

specified schema with a size of 5 could be represented as [11111\*\*\*\*\*...] where unspecified members are denoted by “\*”

A given bit-string  $x$  is an instance of a specific schema  $s$ ,  $x \in s$  if  $x$  matches  $s$  in the defined positions within that schema.  $o(s_i)$  defines the order of  $s_i$  which is the number of defined bits in  $s_i$ . The royal road function was designed to “capture one landscape feature of particular relevance to GAs: the presence of fit low-order building blocks that recombine to produce fitter, higher-order building blocks” [13].

**The Trap-5** concatenated problem is designed to be intentionally deceptive [14,15], such that they “deceive evolutionary algorithms into converging on a local optimum. This is particularly a problem for algorithms which do not consider interactions between variables.” [16]. As with the Royal Road problem, the bit-strings are partitioned into blocks and their fitness scored separately. Seen in Eq. 5a is the function of a trap of order  $k$ .

$$f(x) = \sum_{i=1}^{n/k} \text{trap}_k(x_{b_{i,1}} + \dots + x_{b_{i,k}}) \quad (5a)$$

$$\text{trap}_k(u) = \left\{ \begin{array}{ll} f_{\text{high}} & \text{if } u = k, \\ f_{\text{low}} - u \frac{f_{\text{low}}}{k-1} & \text{otherwise} \end{array} \right\} \quad (5b)$$

Blocks within the bit-string are scored according to the fitness function in Eq. 5b. A Trap5 problem with a bit-string length of 10 would have the values  $n = 10$ ,  $k = 5$ ,  $f_{\text{high}} = 5$  and  $f_{\text{low}} = 4$ .

The further from the goal of each Trap containing five 1’s, the higher the fitness value, with only a maximum achieved when the whole Trap is comprised of 1s, leading the algorithm away from the optimal value.

### 2.3 Principal Components Analysis

The process of reducing the dimensionality of the algorithm trajectory population datasets is done through the use of Principal Components Analysis (PCA). This allows us to project the higher dimensional space of the solutions to a three-dimensional space as “PCA produces linear combinations of the original variables to generate the axes, also known as principal components, or PCs.” [17]. This involves the calculation of a series of perpendicular, non correlated, linear combinations of the variables in the population such that each combination accounts for the maximum possible variation in the dataset through the use of singular value decomposition (SVD). A summary of the calculation of linear combination and weights from [17] can be seen below in the following series of Eqs. 6

$$\begin{aligned} PC_1 &= a_{11}X_1 + a_{12}X_2 + \dots a_{1p}X_p \\ PC_2 &= a_{21}X_1 + a_{22}X_2 + \dots a_{2p}X_p \\ PC_1 &= a_1^t X \\ PC &= XA \end{aligned} \quad (6)$$

In Eq. 6, matrix  $\mathbf{A}$  denotes the matrix of eigenvectors. These are used to show the relationship between the original variables and the orientation of the principal components. The resulting datasets were then mined with the intent of finding features capable of explaining aspects of the optimisation problems that they were generated by.

### 3 Feature Extraction

#### 3.1 Existing Population Diversity Measures

There exist several metrics used to measure the change in population diversity over the course of an optimisation run by genetic-based algorithms. In [18] a brief review of many of these metrics can be found. The metrics covered include the Hamming Distance, the sum of pair-wise comparison in the number of variable differences between two solutions although this method can be considered computationally expensive. An alternative to Hamming Distance is the Moment of Inertia [19] which provides a “..single method of computing population diversity that is computationally more efficient than normal pair-wise diversity measures for medium and large sized EA problems.” When researching possible metrics for comparison, it was decided that the Kullback-Leibler Entropic Divergence distance measure [20] would be the best candidate as it was suitable for both population diversity monitoring and the detection of the “phase transition” point, in which it is said that a population based algorithm moves from the exploration of the search space to the exploitation of known problem structure to generate higher fitness solutions.

**Entropic Divergence.** The Kullback-Leibler Entropic Divergence ( $KL_d$ ) is a population diversity distance measure based around the concept of information gain and Shannon’s Entropy [22] in which “...the entropy of a random variable is defined in terms of its probability distribution.” [20]. It can be defined in the following Equation:

$$KL_d(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left( \frac{P^{(t)}(x)}{Q^{(t_0)}(x)} \right) \quad (7)$$

Where  $P$  and  $Q$  are vectors of marginal probabilities for two different populations in the trajectory [21].

Using the above Equation it is possible to track the information gain from the initial population generated by the algorithms as  $Q(x)$  remains constant as the probability vector of the generation  $t=0$ . This metric is called the “Global Learning” and it measures the total information gain from initial population to the population at any given  $t$ . The expected behaviour for this metric is to increase over time until a “steady state” is arrived at.

It is also shown in [20] that it is possible to use the above  $KL_d$  Equation to measure the information gain between two consecutive populations where  $Q^{(i)}(x)$

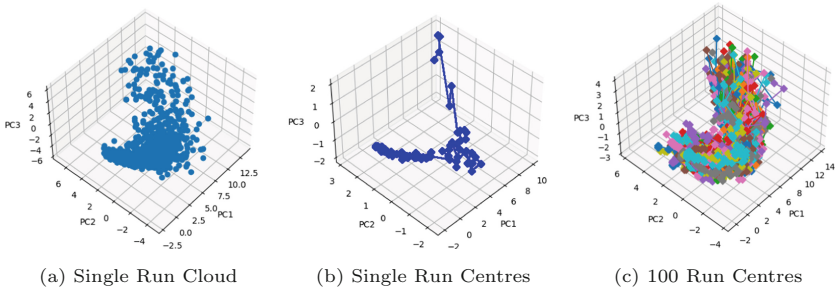
and  $P^{(i+1)}(x)$  are used. This is known as “Local Learning” with the expected behaviour of increasing until a “Phase Transition” point at which the algorithm moves from exploring the search space to exploiting knowledge learned. In the exploitation phase, higher fitness solutions are generated using this implicit knowledge. When this happens it is expected that the local learning rate will decrease as the diversity within the population decreases until convergence has been completed or a local basin of attraction is escaped [23]. This is of interest as it can be used to inform end-users when maximum population diversity is reached in a trajectory.

### 3.2 Sub-space Derived Features

**Population Cluster Centers.** The dimensionality of the dataset is reduced through the projection of the data into a lower dimension set based on the principal components calculated using 6. In this paper we project into a three-dimensional sub-space to help visualise the population as a cluster, illustrated in Fig. 2.a. The centroid of this cluster can be found by calculating the point that minimizes the sum of squared Euclidean distances between itself and each point in the set as seen in Eq. 8.

$$C = \frac{\mathbf{x}_1 + \mathbf{x}_2 + \cdots + \mathbf{x}_k}{k} \quad (8)$$

Figure 2.a is an example of a single trajectory visualisation post-PCA conversion. Each point in the trajectory represents the centroid of a population of solutions. For each generation in a given trajectory the centre point of the cluster is calculated. This process results in a set of points in 3D space that represents the algorithm trajectory, Fig. 2.b, from the initial population to the final population in terms of variation, as measured by the reduction in PCA coefficients over time from  $t = 0$  to  $t = \text{final}$ .



**Fig. 2.** PBIL 1D checkerboard trajectory visualisation

It is important to note that this method does not chart the algorithm trajectory in objective space and does not explicitly reflect the fitness landscape

but instead can be used to measure the direction and magnitude of changes in population diversity after being projected into this subspace. Seen in Fig. 2.c are all 100 trajectories created by the PBIL on the 1D Checkerboard problem, projected against the first three principle components.

**Angle from Origin** measures the angle between the centroid of the initial starting population in the trajectory and each subsequent population that was created. Each of the two points in the space are represented by the centroids coordinates as a vector of [PC1, PC2, PC3] coefficients in place of x, y and z coordinates. In order to calculate the acute angle  $\alpha$  between two vectors we use the inverse cosine of vector products as seen in Eq. 9

$$\alpha = \arccos\left(\frac{C_0 \cdot C_i}{\|C_0\| \|C_i\|}\right) \quad (9)$$

$C_0, C_i = \text{Cluster Centroids (x, y, z)}$

**Angle Between Clusters** is calculated as in Eq. 9 using  $C_i$  and  $C_{i+1}$ , where ( $i <= 0 <= \text{maxGen}$ ). This allows for the angle between consecutive populations to be calculated.

**PCA Loading Values** can be calculated using the resulting matrices from the principal component decomposition process outlined earlier in this paper. Loadings can be considered the weighting of each variable as they describe the magnitude of contribution each variable has to the calculation of each Principal component. Loading signs indicate the type of correlation between the PC and the variable in terms of negative and positive correlation and the strength of that relationship can be seen in the values – larger values indicate a stronger relationship. These loadings are shown in Eq. 6 as the matrix **A** and are the coefficients of the principal components (eigenvectors) with respect to the solution variables.

## 4 Results

We hypothesize that it is possible to derive features from algorithm trajectories that can aid in generating explanations for end-users similar in nature to existing known metrics such as the Kullback-Leibler Entropic Divergence values. For two population-based NDAs – a genetic algorithm and a univariate population based incremental learner – we generated a total of 100 algorithm trajectories on each of the three test functions used. These trajectories were transformed using PCA to allow the projection of the populations into a lower dimension space for the purpose of visualisation and feature extraction.



#### 4.1 PCA Explained Variation

The values in Table 3 show the mean percentage of variation in the population data explained by the first three principal components, broken down by algorithm and problem.

**Table 3.** PCA variance explained by three components

Algorithm	Problem	PCA1 Exp %	PCA2 Exp %	PCA3 Exp %	Total %
PBIL	1D checker	25.6	5.2	3.6	34.4
GA	1D checker	32.5	10.8	7.6	50.9
PBIL	Royal road	25.1	5.9	3.8	34.8
GA	Royal road	28.8	10.0	7.4	46.2
PBIL	Trap5	27.0	4.6	3.3	34.9
GA	Trap5	37.9	11.0	7.6	56.6

These results show that for the PBIL, total variation explained by the first three principal components was 34.4% in the 1D Checkerboard problem, 34.8% in the Royal Road problem and 34.9% in the Trap5 Problem. The results also show that for the GA, explained variation was 50.9% in the 1D Checkerboard, 46.2 in the Royal Road and 56.6% in the Trap5 Problem.

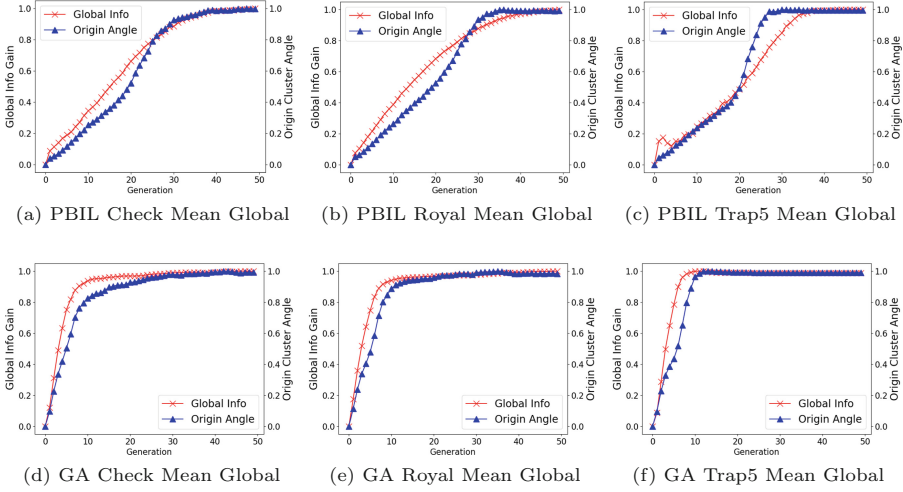
#### 4.2 Information Gain and Cluster Angle Results

Table 4 displays the Spearman Correlation Coefficients of Local and Global information gain to the Inter-Cluster and Angle from Origin features extracted. Global Information shows a strong positive correlation to the Angle from Origin feature with a range of 0.76 to 0.99 across all problems and algorithms. The PBIL coefficients were 0.99 for the 1D Checkerboard, 0.96 for the Royal Road and 0.88 for the Trap5. The GA coefficients were 0.98 for the 1d Checkerboard, 0.88 for the Royal Road and 0.76 for the Trap5 problem.

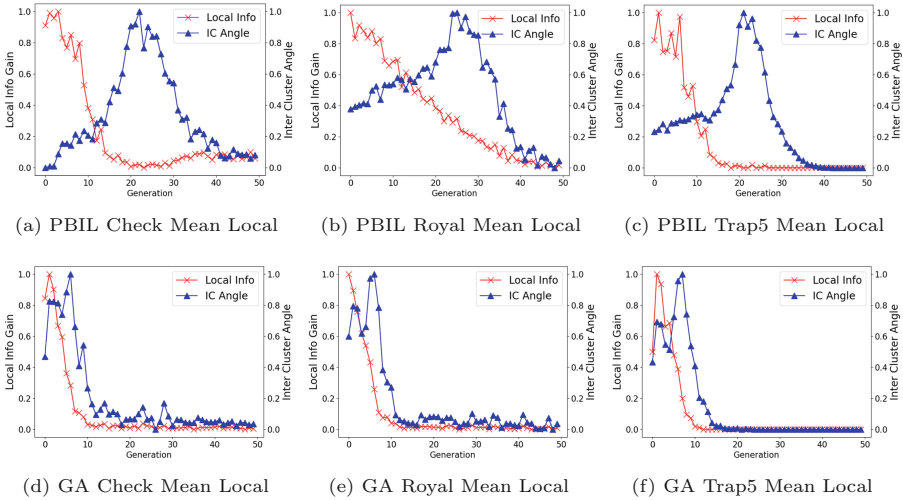
**Table 4.** Spearman Correlation Coefficient

Algorithm	Problem	Local to Inter-Cluster	Global to Origin
PBIL	1D checker	-0.69	0.99
GA	1D checker	0.83	0.98
PBIL	Royal road	0.36	0.96
GA	Royal road	0.94	0.88
PBIL	Trap5	0.39	0.88
GA	Trap5	0.79	0.76

**Global Information Gain and Angle from Origin** comparison results are shown in Fig. 3, split by algorithm and problem. It can be seen in the results and the correlation coefficients in Table 4 that for all three problems and both algorithms, the angle from origin metric closely matches the behaviour of the Global Information Gain behaviour. Both metrics detect the increase in information gained as the algorithms solve the supplied problem.



**Fig. 3.** Global information Vs PCA angles by problem and algorithm



**Fig. 4.** Local information Vs PCA angles by problem and algorithm

**Local information Gain and Inter-Cluster Angle** comparison results are more varied and appear to be showing that learning behaviour differs between algorithms on the same problem, seen in Fig. 4. The inter-cluster angle calculated for the populations generated by the PBIL do not share the same pattern of behaviour as the Local Information gain. The results show a peak approximately 25 to 30 generations later than the local information gain and so these events do not co-occur at the same point in the trajectory in all problems tested. This difference in behaviour is reflected in the wider range of correlation coefficients calculated.

The results for the GA however do show a similar behaviour with a time lag of approximately 5 generations across all problems tested to the local information gain. Both sets of data peak early in the trajectory with the Inter-Cluster Angle peaking approximately 5 generations after the Local Information metric, displaying that the Inter-Cluster metric is detecting the occurrence of phase transition point only slightly later in the trajectory. The Inter-Cluster metric closely follows the profile of the Local Information as supported by the high positive correlation coefficients in Table 4.

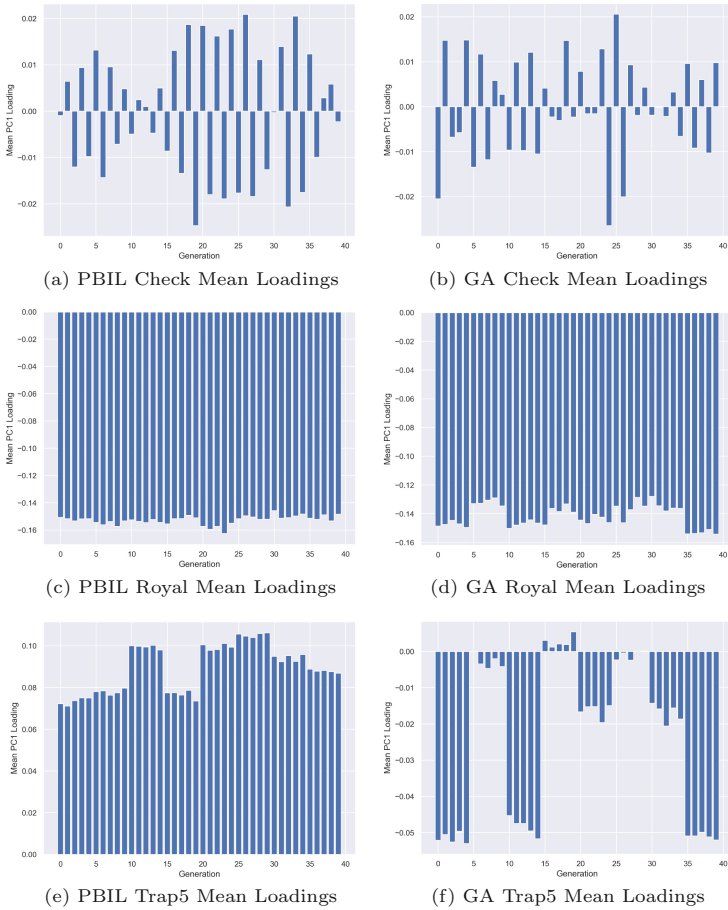
The results show a clear difference between the two algorithms when Local Information Gain is compared to the Inter-Cluster-Angle results. This may be due to the fact that the PBIL increments the probabilistic model gradually over successive populations so local information gain accumulates before it is reflected in Inter-Cluster Angle change. As a GA can be considered a Markov process, the probability of each population is only dependant to the current state of the system. This can also be seen when Global information Gain is compared to Angle from Origin. The PBIL reaches maximum Global Information later in the trajectory than the GA with a shallower ascent. The PBIL reaches point at which Global Information Gain stops increasing between generations 25 and 40 whereas the GA has a steeper Global Information Gain rate, reaching the maximum value between generations 10 and 20. This may be due to the GA taking a more varied path across the search space than the PBIL which tends to have less varied performance. Together, these show that it is possible to detect differences in algorithm behaviour over the same optimisation problems through the differences in both sets of results.

### 4.3 Principal Component Loadings

The results of charting the mean loadings across all runs for each algorithm and problem can seen in Fig. 5.

The 1D Checkerboard results show that the loadings reflect the patterns of the coefficients. Adjacent variables in the solutions discovered have opposing values in both the PBIL Fig. 5.a and GA Fig. 5.b figures for the majority of cases. This matches closely the mathematical structure of the fitness functions. Both algorithms however show instances in which the loadings did not conform to the expected pattern, showing a flip in the alternating sequence at three or more points in the bit-string. The Royal Road results for the PBIL in Fig. 5.c do not show any clear pattern that would match the expected fitness function

structure however the GA in Fig. 5.d does show some partial detection, with consecutive blocks of 5 bits having similar values that do not match the next block in 4 instances. The results for the Trap5 problem for the PBIL in Fig. 5.e do not show any strong relation to the expected fitness function structure however the GA in Fig. 5.f captures this correctly. It shows all 8 blocks of 5 consecutive bits possess similar values but are distinct from the next block. Since PBIL is univariate, it cannot detect multivariate interactions. 1D Checkerboard results show that some bivariate interaction was detected but this will be accidental. These results shows that the algorithm trajectories reflect the simpler features of the problem structure that the algorithms have learned but the higher order features are less likely to be recovered.



**Fig. 5.** PCA loading values by problem and algorithm

## 5 Conclusions

In this paper, we presented the results of the application of Principal Components Analysis (PCA) to the trajectories created by two population-based Non-Deterministic Algorithms (NDA). This was done to mine features that can enrich explanations regarding how these algorithms traverse the search space and present significant solution features detected by the algorithms. We generated a collection of algorithm trajectories by solving a set of benchmark problems with a Genetic Algorithm (GA) and modified Population Based Incremental Learning (PBIL) algorithm and projected the resulting trajectories into a lower dimensional space through the application of PCA. This process resulted in a dataset that was mined using a novel set of angular-based metrics. Our evaluation of these metrics when compared to the Kullback-Leibler Entropic Divergence measure of both Local Information and Global Information gain shows that there is potential to capture a similar level of detail regarding the Global information learned. These metrics were used to detect differing algorithm behaviour on the same problems as seen between that of the PBIL and GA in the Inter-Cluster Angle values. Finally, it was shown that principal component loadings were used to represent what the algorithms have learned in terms of variable contributions to overall fitness. This is a move towards the generation of explanation of solutions returned by the algorithm. This can be seen in the Eigenvector values for the GA that implied the fitness function structure of the optimisation problem for the 1D Checker and Trap 5 Problem. This feature in the PBIL results show partial structure detection only in the 1D Checker problem and shows that some structure has not been captured using the features used in these tests. Being univariate, PBIL is incapable of creating probability features that capture higher level features with interactions as found in the remaining problems. The results of this paper have shown that the PC derived features are associated with the algorithm learnings regarding problem structure. These techniques can be considered a stepping stone towards supporting explanations by relating changes in information gain to the discovery of specific interaction features.

## References

1. Ordish, J., Brigden, T., Hall, A.: Black Box Medicine and Transparency. PHG Foundation, Cambridge, p. 34 (2020)
2. Shakya, S., McCall, J., Brownlee, A., Owusu, G.: DEUM - distribution estimation using markov networks. In: Shakya S., Santana R. (eds) Markov Networks in Evolutionary Computation. Adaptation, Learning, and Optimization, vol. 14, pp 55–71. Springer, Heidelberg (2012). [https://doi.org/10.1007/978-3-642-28900-2\\_4](https://doi.org/10.1007/978-3-642-28900-2_4)
3. Collins, T.D.: Applying software visualization technology to support the use of evolutionary algorithms. *J. Vis. Lang. Comput.* **14**(2), 123–150 (2003). ISSN 1045–926X. [https://doi.org/10.1016/S1045-926X\(02\)00060-5](https://doi.org/10.1016/S1045-926X(02)00060-5)
4. Ochoa, G., Malan, K.M., Blum, C.: Search trajectory networks: a tool for analysing and visualising the behaviour of metaheuristics. *Appl. Soft Comput.* **109**, 107492 (2021). ISSN 1568–4946. <https://doi.org/10.1016/j.asoc.2021.107492>

5. Pohlheim, H.: Multidimensional scaling for evolutionary algorithms-visualization of the path through search space and solution space using Sammon mapping. *Artif. Life* **12**(2), 203–209 (2006). PMID: 16539764. <https://doi.org/10.1162/106454606776073305>
6. Michalak, K.: Low-dimensional Euclidean embedding for visualization of search spaces in combinatorial optimization. *IEEE Trans. Evol. Comput.* **23**(2), 232–246 (2019). <https://doi.org/10.1109/TEVC.2018.2846636>
7. Baluja, S., Caruana, R.: Removing the genetics from the standard genetic algorithm. In: *ICML*, pp. 38–46 (1995)
8. Baluja, S.: An empirical comparison of seven iterative and evolutionary function optimization heuristics, Carnegie Mellon University, Pittsburgh, PA, Technical report CMU-CS-95-193 (1995)
9. Holland, J.H.: *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Oxford. U Michigan Press, England (1975)
10. Goldsmiths University of London Computational Creativity Research Group. [http://ccg.doc.gold.ac.uk/ccg\\_old/teaching/artificial\\_intelligence/lecture16.html](http://ccg.doc.gold.ac.uk/ccg_old/teaching/artificial_intelligence/lecture16.html). Accessed 12 Nov 2020
11. Baluja, S., Davies, S.: Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. Technical report, DTIC Document (1997)
12. Forrest, S., Mitchell, M.: Relative building-block fitness and the building block hypothesis. In: *Foundations of Genetic Algorithms 2* (San Mateo), Morgan Kaufmann, pp. 109–126 (1993)
13. B.2.7.5: Fitness Landscapes: Royal Road Functions. *Handbook of Evolutionary Computation* M MitchellS Forrest
14. Goldberg, D.E.: Genetic algorithms and Walsh functions: part i, a gentle introduction. *Complex Syst.* **3**(2), 129–152 (1989)
15. Goldberg, D.E.: Genetic algorithms and Walsh functions: part ii, deception and its analysis. *Complex Syst.* **3**(2), 153–171 (1989)
16. Brownlee, A.E.I.: Multivariate Markov networks for fitness modelling in an estimation of distribution algorithm. Robert Gordon University, PhD thesis (2009)
17. Holland, S.M.: *Principal Components Analysis (PCA)* (2019). Strata.uga.edu. <https://strata.uga.edu/software/pdf/pcaTutorial.pdf>. Accessed 19 Jun 2021
18. Hien, N.T., Hoai, N.X.: A Brief Overview of Population Diversity Measures in Genetic Programming (2006). <http://gpbib.cs.ucl.ac.uk/aspgp06/diversityMeasures.pdf>. Accessed 20 Jun 2021
19. Morrison, R.W., De Jong, K.A.: Measurement of population diversity. In: Collet, P., Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M. (eds.) *EA 2001. LNCS*, vol. 2310, pp. 31–41. Springer, Heidelberg (2002). [https://doi.org/10.1007/3-540-46033-0\\_3](https://doi.org/10.1007/3-540-46033-0_3)
20. Cutello, V., Nicosia, G., Pavone, M., Stracquadanio, G.: Entropic divergence for population based optimization algorithms. In: *IEEE Congress on Evolutionary Computation*, pp. 1–8 (2010). <https://doi.org/10.1109/CEC.2010.5586044>
21. MacKay, D.J.C.: *Information Theory, Inference, and Learning Algorithms* (First ed.). Cambridge University Press, p. 34 (2003). ISBN 9780521642989
22. Shannon, C.E.: A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.* **5**(1), 3–55 (2001)
23. Protter, M.H., Morrey, Jr., Charles, B.: *College Calculus with Analytic Geometry* (2nd ed.) (1970)