PCA Landscape Projections for Meta-Analysis of Genetic Improvement for Source Code

Zsolt Németh[1][0000−0001−5753−8299], Penn Faulkner Rainford[2][0000−0002−0552−2209], and Barry Porter[1][0000−0001−8376−736X]

[1] School of Computing and Communications, Lancaster University, UK
{z.nemeth, b.f.porter}@lancaster.ac.uk
[2] Research IT Services, University of York, UK
penn.rainford@york.ac.uk

Abstract. Genetic improvement (GI) for source code uses evolutionary methods to improve either the functional or non-functional elements of existing software. We consider GI for performance improvement, which has notably different requirements to traditional bug-fixing applications of GI: in performance improvement we mix together genetic programming (synthesis of new material) with genetic improvement, and the extent to which the genotype and phenotype landscapes are aligned (together with the mutation grammar) determines the success of a GI search. Because GI process are inherently complex and have a wide range of tunable parameters, gaining a clear understanding of which GI mechanisms lead to which outcomes is of crucial importance in identifying successful GI mechanisms for a given problem type. We introduce a PCA-based landscape analysis method to gain a detailed understanding of what happens during a GI execution, both at the level of genetic individuals and of entire experiments under various configurations.

Keywords: Genetic Improvement · Fitness Landscapes · Performance Optimisation.

1 Introduction

Genetic improvement for source code [15] seeks to use evolutionary methods to enhance either functional or non-functional elements of an existing program. Common applications include automated bug fixing [9] and performance optimisation [8]. In this paper we focus on GI for performance optimisation, which has a very different problem formulation to that of bug fixing. In GI for performance optimisation of source code, the level of connection between the genotype landscape and the phenotype landscape – together with with the specific fitness function that is chosen – generally determines the success of a GI search. GI for performance optimisation requires a mixed approach between genetic improvement and genetic programming, where new genetic material can be synthesised in order to help control the trajectory of the search over the fitness landscape, along with the re-use and adaptation of existing genetic material.

The way in which GI processes really work in this problem domain remains relatively poorly understood. While fitness landscape [14] is a fundamental concept in GI, a simple genotype-phenotype fitness function is unable to reveal the fine details of the evolutionary process or the search space. There have been proposals to augment fitness with additional information such as the presence of synergy and antagonism in a GI for an energy efficiency scenario [1], using *landscape basin depths* over *autocorrelation of the fitness values* of random walks for characterising difficulty [6], introducing dual, functional (number of test cases passed) and non-functional (execution time) landscape [4], or use visual analysis to capture novelty and quality-diversity aspects of search [16]. All these approaches are tailored to a specific scenario and its requirements. In our experience monitoring additional parameters can add further levels of complexity while the meaningful information remains hidden or gets even more obfuscated. We propose a well known generic statistical method, Principal Component Analysis (PCA), that has been applied successfully in many other fields [5], for reducing the complexity of high-dimensional data and extracting the attributes that carry the characteristic information. In this paper we report on a study to reveal the effects of evolutionary mechanisms on observed attributes of individuals and populations and the degree to which these indicators are sensitive or indifferent to experimental conditions. We then use landscape projections on top of our PCA data to situate those correlations within different elements of a GI process, such as the individuals present in the first and last generations.

We show the effects of this method when applied to GI for performance improvement of two different applications: a hash table and a cache. We demonstrate both the effects on individuals of a GI run, and the effects across different GI configurations when observed at the whole-experiment level across many different experiments. Our results demonstrate that our PCA landscape projection method accurately captures and visualises multiple dimensions of metrics and their distribution across a two-dimensional space.

2 Methodology

We use a genetic improvement (GI) framework which is designed to enhance the performance of an isolated software component, such as a hash table or a cache. We use selected sequences of function calls into those components and attempt to find performance-enhanced versions which are faster at dealing with those particular call sequences by specialising elements of their implementations (while also maintaining functional correctness). In each case we narrow the search to improving one or two hand-selected functions of those components (such as the hash function of a hash table, or the eviction policy of a cache).

Our GI system operates on the source code itself in alternate rounds of mutation and crossover, allowing us to separate out the effects of each modification type: after one generation we apply mutations and selection, then for the next generation we apply crossover and selection, then back to mutation, and so on. Our mutation grammar includes the synthesis of new lines of code (such as variable declarations, conditional branching, and assignments), plus mutations or

deletions of existing lines of code in a way that is designed to always result in syntactically and semantically correct individuals. Our crossover strategy is one-way single-point horizontal gene transfer, where we inject one line of code from one individual into a randomly-chosen semantically-valid point in another individual; if that chosen line is a scope header (such as a loop header) then the attached scope is also injected in the crossover.

For the context of this paper we apply our GI framework to two application domains: a hash table and a cache. For the hash table, the GI system attempts to find a (source code for a) hash function which distributes keys across hash buckets in such a way that retrieval speed is maximised for a particular sequence of put and get calls: this may either be by maximising the uniformity of the distribution to the set of keys given, or by skewing that distribution towards the request frequency of each key. For the cache, the GI system attempts to find an eviction policy which minimises cache misses for a given request sequence; this is achieved by boiling down the cache policy to two functions; considering an ordered list of cache items where we always evict from one end: (1) a function where to place an item on initial insertion into the cache, and (2) a function where to move an item to when it is accessed.

Based on prior research on classifying individuals by selected phenotypic features, hence the notion of species and speciation plays a pivotal role in our model as introduced in [13]. We can distinguish two ways of performance improvement: (i) better, more efficient realisation of an algorithm and (ii) algorithmically (semantically) different solution to the problem. Results showed that semantic changes not only introduce better improvement than text rewritings but overall, they are essential to maintain diversity and utility in the population [12]. Our definition of species is a group of programs (source code) that are semantically equivalent, realise the exact same algorithm. Capturing the semantic equivalence is not trivial and application dependent. We devised the species metrics for hash algorithms as the Kullback-Leibler divergence [7] on the hash layout for a known training set, and for cache algorithms, as the miss ratio. The concept of species is used in this study but not in scope of the present paper.

2.1 Principal Component Analysis

Principal Component Analysis (PCA) [3] is a well-known statistical method to reduce the dimensionality of datasets. It does this by suppressing those dimensions that are less distinctive, i.e. where variance is small, and establishes principal components, a linear combination of the initial spanning vectors, to maximise the distance between data points. The method assumes a data matrix of n rows of data items (samples) and k columns of variables; each data item is represented by a vector of these k variables $x_i = [v_{i1}, v_{i2}, \ldots v_{ik}]$. First, the data are normalised, i.e. column-wise the displacement of variables from their mean value are taken and then divided by their (biased) standard deviation, $v'_{ij} = \frac{v_{ij} - \mu_j}{\sigma_j}$. This step brings variables of various orders of magnitude to the same scale. Then there are two pathways: (1) Singular Value Decomposition

(SVD) [17] on the normalised data, or (2) calculate the $k \times k$ correlation matrix of the normalised data and apply Eigenvalue Decomposition (EVD) [17] to the matrix. In case of SVD the resulted k (right) singular vectors span the new space whereas in case of EVD, they are the k eigenvectors. These vectors, termed component or loading vectors, are associated with an *explained variance* factor, showing how much the corresponding vector contributed to the total variance. Since many component vectors contribute negligibly, they can be simply omitted without compromising the accuracy of the data. The resulting space, with reduced dimensionality, exaggerates the differences and suppresses the similar features.

2.2 Metrics

Hash example. In this case we observed 6 metrics: execution time (t) as fitness, the Kullback-Leibler divergence (D_{KL}) as a species indicator, the individual's rank within the population (R), the relative fitness compared to the progenitor individual (F_r), and the change in fitness (Δ_f) and the change in rank (Δ_r) with respect to those of the parent individual. Table 1 shows the correlation values between the metrics. t and D_{KL} are direct measurements and are absolute values for the individual; the others are computed values: R, F_r, Δ_f and Δ_r characterise an individual's position relative to the rest of the population, the progenitor individual and its own parent, respectively. Although a large number of parameters are recorded, this procedure introduces minimal intrusion and complexity as D_{KL} is measured outside the timed section and all relative metrics are calculated off-line in a post-mortem way. These metrics are meant to capture many details of phylogenetics, i.e. the evolutionary relationship between individuals of successive generations, but at the same time their number and intricate interplay make it hard to extract meaningful information. To illustrate the problem, let us consider two individuals in an experiment captured as: $\{t = 48342, D_{KL} = 3.36, R = 3, F_r = 0.45, \Delta_f = 0.45, \Delta_r = -9\}$ and $\{t = 30678, D_{KL} = 0.23, R = 37, F_r = 0.28, \Delta_f = 1.13, \Delta_r = 24\}$. At first sight, the latter has better fitness and a very impressive relative fitness, showing over 70% of improvement. It is, however, ranked as 37th out of 50, dropping from 13th (as opposed to 3rd position of the other one, raised from 12th) and is actually slower by 13% than its parent. Additionally, in [13] the fitness landscape and the optimum spots of the species within the landscape were analysed; $D_{KL} = 0.23$ in the present case suggests that potentially the individual is over the the optimum range of the species metrics. Apart from the execution time, based on the metrics alone, it is hard to compare which individual belongs to a position higher on the phylogeny or has better potential for further improvement.

Cache example. In the case of cache performance improvement, altogether 12 metrics are captured for each individual: adjusted time ($\hat{t}$), miss ratio (P_m), execution time (t), penalty time (t_p), put parameter (n_{put}), update parameter (n_{upd}), cache hits (H), cache misses (M), and the metrics already introduced in the hash scenario: rank within the population (R), the relative fitness (F_r), the

	t	D_{KL}	R	F_r	Δ_f	Δ_r
t	1	0.765	0.456	0.996	0.773	-0.334
D_{KL}	0.765	1	0.511	0.762	0.534	-0.343
R	0.456	0.511	1	0.456	0.346	-0.493
F_r	0.996	0.762	0.456	1	0.778	-0.333
Δ_f	0.773	0.534	0.346	0.778	1	-0.422
Δ_r	-0.334	-0.343	-0.493	-0.333	-0.422	1

Table 1: The correlation matrix of the metrics in the hash experiments.

change in fitness and rank (Δ_f, Δ_r) from the antecedent generation. From these t, n_{put}, n_{upd}, H and M are observed and recorded at run-time. n_{put} and n_{upd} are the parameters computed by the two functions that determine where to place an item on initial insertion, and where to move it when accessed, respectively, cf. Sec. 2. These two variables are the output of the code being optimised, hence they serve as a direct indicator of the status of the procedure. It is ambivalent what metrics could characterise the performance of a cache policy. Strictly speaking, it is the miss ratio $P_m = M/(H+M)$, but it does not take into consideration the time necessary to traverse the data structure, calculate the positions and insert the new elements into the cache. Therefore, an alternative performance indicator, adjusted time $\hat{t} = t + t_p$ is introduced, where the running time t captures all the aspects mentioned above and is adjusted with the penalty time t_p proportional to the number of misses $t_p \propto M$, emulating the higher cost of accessing data out of cache. Accordingly, there are two disjunct versions for these experiments: one where adjusted time ($\hat{t}$) acts as the fitness metrics and miss ratio (P_m) as species indicator; and another one, where both the fitness metrics and species indicator is the miss ratio (P_m.) The meaning and calculation of R, F_r, Δ_f and Δ_r are as in the previous example.

3 Results

We demonstrate the utility of PCA at two levels: (i) experiments running under the same circumstances, where the metrics of *individuals* are observed and analysed so that some of the features of their phylogeny are highlighted; (ii) experiments running in different conditions, where the metrics of *collective experiment sets* are observed to quantify the effects of those differences.

3.1 The evolution of a hash function

In this section we present the observations we obtained during a single configuration of an experiment for improving the performance of a *hash function*. There were 50 individuals in a generation and the improvement ran for 40 generations, with the GI framework configured to use *weighted selection* with *2 elites*. We recorded the 6 parameters explained in Sec. 2.2; the experiment was executed several hundred times and the collected data were collated for 100 sampled runs. Applying the (correlation EVD) PCA to the set

of data, the resulting primary components (also called loading vectors, technically eigenvectors in this case) are in Table 2a. The explained variance vector is $[0.644, 0.163, 0.102, 0.059, 0.031, 0.0006]$, hence the first two eigenvectors, shown in the table, are the two most dominant components and contribute to over 80% of the total variance. Thus, the search space in the following charts on hash experiments is spanned by

$$PCA_1 = [0.478 * t, 0.425 * D_{KL}, 0.327 * R, 0.478 * F_r, 0.419 * \Delta_f, -0.281 * \Delta_r]$$

and

$$PCA_2 = [-0.281 * t, -0.088 * D_{KL}, 0.554 * R, -0.283 * F_r, -0.171 * \Delta_f, -0.705 * \Delta_r]$$

	t	D_{KL}	R	F_r	Δ_f	Δ_r
0	0.478	0.425	0.327	0.478	0.419	-0.281
1	-0.281	-0.088	0.554	-0.283	-0.171	-0.705

(a) 100 runs of a single configuration of hash experiments (weighted selection, 2 elites.)

	t	D_{KL}	R	F_r	Δ_f	Δ_r
0	0.489	0.455	0.375	0.491	0.349	-0.224
1	0.323	0.196	-0.386	0.316	-0.268	0.732

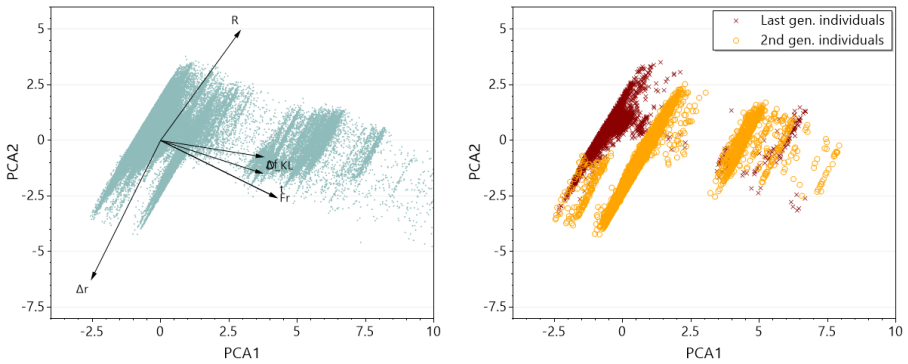
(b) Collated data of 12 configurations of hash experiments, 3 selection strategies × 4 elite categories, 100 runs each.

Table 2: The resulting primary components of the EVD PCA applied to all individuals of the collated experiments.

The biplot [2] in Fig. 1a simultaneously presents all the individuals of the evolutionary processes in the reduced, two-dimensional space spanned by PCA_1 and PCA_2, with the landscape directionality of our six metrics overlaid as a semantic aid. More precisely, these six metrics are shown as vectors of their relative contribution to the principal components [3]. They can give good qualitative indications about the directions and tendencies in this space. For instance, at first glance Fig. 1a does not show more than 3 major clusters and many dispersed individuals. Yet, the t and the almost coinciding F_r vectors show a slightly diagonal line where higher running times/poorer relative fitnesses are on the right hand side and better running times/better relative fitnesses are on the left hand side; the 0 point on all axes represents the mean. We can therefore conclude that the 3 major clusters are 3 *different* performance groups. Furthermore, a fine striping, perpendicular to the t, F_r axis in the population pattern can be observed, as an uneven, quantised distribution along the performance axis. The almost coaxial but inverted R and Δ_r vectors are orthogonal to t and F_r, indicating that individuals in a performance stripe differ mainly in their ranks and changes in ranks. Evidently, R and Δ_r are in opposing directions: individuals with good (low) rank have less chance to improve (negative Δ_r); consider the extremities: the very best individual can only worsen (or maintain) its rank whereas the very worst one can only improve.

We can further augment the data points with markings according to some other parameters, making the chart virtually 3 dimensional. Marking the second[1]

[1] The starting individuals in the first generation are identical to one another, such that PCA technically does not work when applied to this generation, whereas the second



(a) Biplot of all individuals and the 6 metrics: t, D_{KL}, R, F_r, Δ_f and Δ_r.

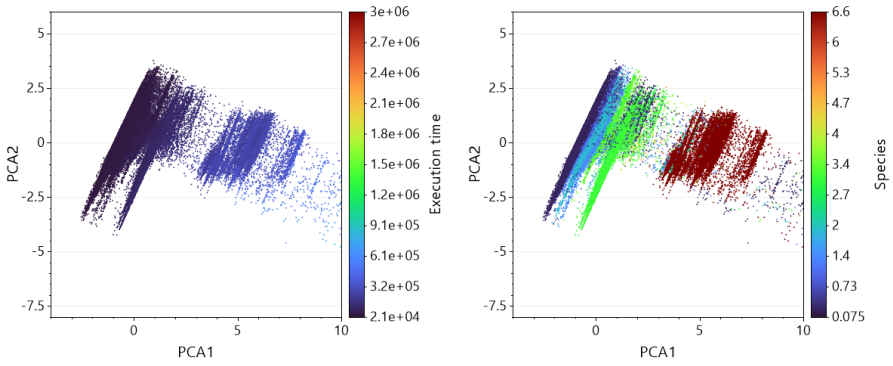
(b) Individuals of an early (after one round of mutations) and the last generations.

Fig. 1: All individuals of 100 hash experiments in the transformed search space.

and last generations of the evolution in Fig. 1b, shows nearly disjunct groups of early and later individuals, indicating that most of their lineages are traversing the search space and are different at the end of the evolution according to the principal components. Individuals are layered along the F_r and t axis showing different performance groups, and spread out quite naturally along the R and Δ_r (perpendicular to the performance related F_r and t): within similar performance their ranks and changes in rank do necessarily differ. In fact, wide stripes along the R axis means that *most ranks* belong to the same performance group hence, the very wide red patch means that at the end of the evolution most individuals within a population are at the best (leftmost on the F_r and t axis) performance level. It is also shown that the last generation individuals have better performance than the earlier individuals, demonstrating that the genetic process does achieve performance improvement. In Fig. 2a the execution time is added as the third dimension above the plane of the two dominant components. The gradient is visible, corresponding to the t vector of Fig. 1a; the very best running times are towards the $(0, 0)$ region in alignment with the findings in Fig. 1b, but otherwise the fitness metrics does not reveal anything about the nature of the genetic improvement. Finally, Fig. 2b shows the species metrics for the same distribution of individuals. The D_{KL} axis is similar to t but not identical; fitness and speciation are not a tautology in this instance. While there is a clear correlation between species and fitness, speciation is able to establish fuzzy bordered clusters of individuals. This is due to the fact that speciation establishes the potential of an individual, the boundary of its fitness e.g. best possible fitness, but actual metrics may diverge; the same conclusions were drawn from a different analysis in [13] and the two independent results support each other. Hence, species are clustered (separated by colours) but some fuzziness and overlap of these clusters

generation has already had a round of mutations from which to draw statistical inferences.

is possible and inevitable (colours are slightly scattered.) Here it can be seen that the Kullback-Leibler divergence has the ability to separate the individuals and establish regions in the search space for the particular problem of hashing. This makes it a key property to realise strategies for increasing diversity or searching for genuine novelty – based on whether or not new individuals belong to a new species, rather than simply having a different genotype.



(a) The execution time t in μs (primary fitness metrics) of the individuals.

(b) The species metrics (Kullback-Leibler divergence, D_{KL}) of the individuals.

Fig. 2: All individuals of 100 hash experiments, spatial distribution of speed and species metrics.

3.2 Comparison of experiments

As shown above, PCA analysis has the capability to reveal some details present in the recorded data of a GI experiment, but otherwise hidden in a simple direct analysis. In this section we demonstrate what PCA can add when comparing experiments across different conditions.

The experiments. We set up 12 configurations of experiments with varying settings for the GI framework, which control how its selection strategy and history-tracking work, while all other parameters were identical. The selection strategy is related to how offsprings are chosen for the next generation. Here we implemented (a) random selection, (b) weighted selection, where the probability of getting to the next generation is proportional to the *fitness distribution* within the population, and (c) ranked selection, where the same probability is proportional to the *rank* within the population. Ranked selection tends to exaggerate the smallest differences in fitness and diminish the largest differences. The history aspect, most often realised as elitism, is the mechanism by which successful individuals from previous generations are preserved and taken into consideration at subsequent selection rounds. Here we implemented strategies with (i) 0 elites, (ii) 2 elites (this is the standard in our experiments), and two versions (iii, iv) of

dynamic (ageing) elites where elites are brought forward not just one generation but 1 or 2 additional ones; in the latter case the effective number of elites in a population may change dynamically between 2 and 6. Additionally, in the case of cache experiments, the progenitor can be placed at two points of the fitness landscape when we begin the experiment (constructed and evaluated by simulated parameter studies): position A, an 'average' point in the landscape known to be equally far from the best and worst possible fitness values; and position B the worst-possible starting position in the search space.

When we compare these strategies using elementary measures, we find that the results are ambiguous, with nearly all of them able to produce the same improvement at the last generation. The simple comparison of the fitness, the improvement of fitness, or the convergence speed are therefore not adequate to explain potential effects of different configurations. It is, however, possible to join together the results of experiments of different configurations and establish a common PCA-coordinate system while keeping the third dimension for separating the configurations of experiments by colours. The resulting coordinate system would be different from any of the coordinate systems of the individual experiments, e.g. Table 2b of the eigenvectors of all individuals of the 12 configurations.

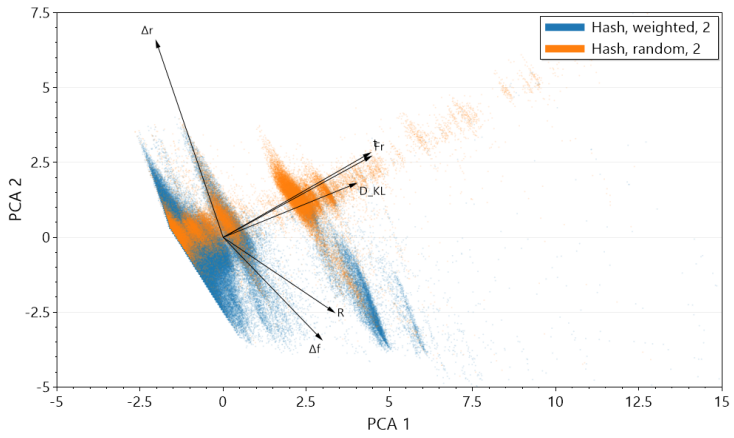


Fig. 3: Comparison of two configurations of hash experiments: weighted vs random selection. Both use execution time t as fitness, with 2 elites.

Fig. 3 compares two configurations of hash experiments: random vs weighted selection. The t axis determines the performance (fitness) gradient; it can clearly be seen here that the very best individuals are towards the left (the smaller the better), and are about the same relative fitness if projected onto the F_r axis. However, the *spread* of the very best individuals (along a line perpendicular to the F_r axis) is far more extensive for the weighted selection configuration: the blue area is much wider than the orange one. This implies that weighted selection can produce (or simply keep alive) more individuals that are close to

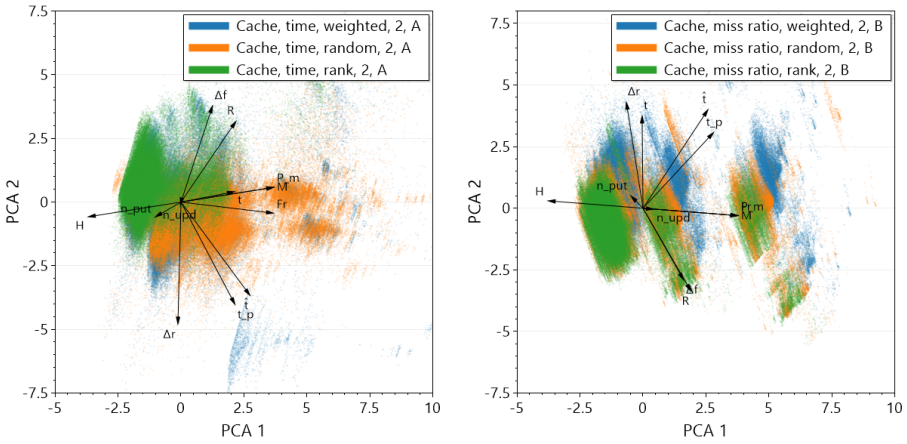
the optimum than random selection. Random selection has a long tail along the t, F_r (and D_{KL}) axis whereas weighted selection has a distinct wing shaped area along the R axis. This suggests that random selection is better at finding new species but is unable to fill all ranks of a generation with individuals of the same performance or species; the orange stripe is narrow with around average rank R and Δ_r. On the other hand, weighted selection has better control not to spread towards the higher values on the t, F_r axis. The blue wing represents individuals with high rank and low performance - unlike the orange area, where individuals can have much lower rank in the same performance region. Since there are no blue individuals of lower rank in this performance stripe, they must be towards the left, better t values. This indicates a simultaneous improvement of performance and ranking in case of weighted selection.

The biplot in Fig. 4a shows the comparison of three configurations of *cache* experiments together with the axes of 12 metrics, as described earlier in Sec. 2.2. The precise analysis of the figure is beyond the scope of this paper; instead we briefly note key selected features. The performance axis here is denoted with $\hat{t}$, where individuals are slower to the lower right direction and quicker to the upper left one. The total time $\hat{t}$ is the algorithmic speed t plus the penalty time t_p for the cache misses. The direction of $\hat{t}$ is closer to t_p; $\hat{t}$ is dominated by the penalty times. The algorithmic speed t is almost orthogonal to either t_p or $\hat{t}$ indicating a low correlation; indeed, the speed of the calculation of positions is independent from the quality of the resulted positions. Misses M and hits H form a 180 degree angle corresponding to their correlation of -1. The two parameters n_{put} and n_{upd} that control the cache placement algorithm, have very little contribution to the overall variation. Similarly to the hash example, R and Δ_r are roughly opposite to each other and perpendicular to F_r. Due to the higher dimension of metrics, their interaction is more complex and not all details can be explained. Towards the better individuals, top left corner, higher H and lower $\hat{t}$, the ranked selection is dominant both in fitness and the coverage along the R axis, i.e. it is able to generate an entire generation of nearly the same high-fitness individuals.

Changing other parameters can alter the whole character of the chart, Fig. 4b. The use of miss ratio P_m as fitness, instead of adjusted time $\hat{t}$, makes all patches crisper and the order of the selection strategies somewhat clearer – as generally rank-based selection is the best and weighted-selection is the weakest. The duplicate orange patches in the centre of Fig. 4a are caused by the progenitor's starting point A; they are not present if started from position B. While the complete analysis of these scenarios are out of scope of the paper, these results demonstrate a powerful way to analyse the internal effects of different genetic improvement mechanisms on the overall trajectory of a search.

4 Discussion and Conclusion

Our use of PCA on data recorded during GI processes has helped to confirm previous hypotheses around speciation in this domain and also revealed some subtle details that direct analysis of the collected data does not show. We can



(a) Execution time t as fitness, 2 elites and initial position A on the landscape.

(b) Miss ratio P_m as fitness, 2 elites and initial position B on the landscape.

Fig. 4: Comparison of three configurations of cache experiments: weighted, random and ranked selection.

identify three main areas of novel findings using this method: (i) The definition of species, i.e. the quantitative classification of algorithmic differences, are correct for the specific hash and cache experiments. Species indicators are application dependent per se, therefore there is no method for establishing species metrics universally. Although our definitions were carefully considered and tests have proven their suitability, a biplot of a PCA analysis enables a simple visual check. (ii) The variable axes on the biplot are a visual representation of the correlation matrix and can give a hint for choosing the right parameters in terms of redundancy, selectivity and sensitivity. (iii) The PCA analysis can distil information that is not immediately obvious from fitness data. Questions about the overall quality of the procedure, such as what proportion of the population can reach a certain performance, or what the chances are of finding good individuals at a given generation step, or how various GI strategies differ and in which ways, can be answered from visual cues.

Overall, these results indicate the potential for highly novel insights into the way in which complex GI processes really work and why, and may lead to new approaches in the search for genetic diversity, novelty, or avoiding neutral areas – elements which result from complex combinations of factors rather than simple fitness or species indicators. In future work, we will complete the thorough analysis of the effect and interplay of parameters and evolutionary strategies just showcased in this paper; to this end we may consider ablation studies and dimension-reduction techniques such as UMAP [11] or t-SNE [10]; and generally, aim to further leverage this analysis approach to help design novel combinations of GI mechanisms – such as elitism – or parameter values which maximise a desired property – such as convergence time or genetic diversity –, for each specific problem area.

Acknowledgments. This work was supported by the Leverhulme Trust Research Grant 'Genetic Improvement for Emergent Software', RPG-2022-109.

References

1. Bruce, B.R., Petke, J., Harman, M., Barr, E.T.: Approximate oracles and synergy in software energy search spaces. IEEE Trans. on Software Eng. **45**(11), 1150–1169 (2019)
2. Gabriel, K.R.: The biplot graphic display of matrices with application to principal component analysis. Biometrika **58**(3), 453–467 (Dec 1971)
3. Greenacre, M., Groenen, P.J.F., Hastie, T., D'Enza, A.I., Markos, A., Tuzhilina, E.: Principal component analysis. Nature Reviews Methods Primers **2**(1) (Dec 2022)
4. Haraldsson, S.O., Woodward, J.R., Brownlee, A.E.I., Smith, A.V., Gudnason, V.: Genetic improvement of runtime and its fitness landscape in a bioinformatics application. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. p. 1521–1528. GECCO '17, ACM (2017)
5. Jolliffe, I.T., Cadima, J.: Principal component analysis: a review and recent developments. Philosophical Trans. of the Royal Society A: Mathematical, Physical and Engineering Sciences **374**(2065) (Apr 2016)
6. Kinnear, K.: Fitness landscapes and difficulty in genetic programming. In: Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence. pp. 142–147 (1994)
7. Kullback, S., Leibler, R.A.: On information and sufficiency. The Annals of Mathematical Statistics **22**(1), 79–86 (1951)
8. Langdon, W.B., Harman, M.: Optimizing existing software with genetic programming. IEEE Trans. on Evol. Comp. **19**(1), 118–135 (2015)
9. Le Goues, C., Nguyen, T., Forrest, S., Weimer, W.: Genprog: A generic method for automatic software repair. IEEE Trans. on Software Eng. **38**(1), 54–72 (2012)
10. van der Maaten, L., Hinton, G.: Visualizing data using t-sne. Journal of Machine Learning Research **9**(86), 2579–2605 (2008)
11. McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction (2020), https://arxiv.org/abs/1802.03426
12. Nemeth, Z., Faulkner Rainford, P., Porter, B.: Reaching meaningful diversity with speciation-novelty in genetic improvement for software. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 1017–1025. GECCO '25 (2025)
13. Nemeth, Z., Faulkner Rainford, P., Porter, B.: Revisiting the fitness landscape of genetic improvement for source code: A phenotypic speciation approach. ACM Trans. Evol. Learn. Optim. (Jul 2025)
14. Petke, J., Alexander, B., Barr, E.T., Brownlee, A.E.I., Wagner, M., White, D.R.: A survey of genetic improvement search spaces. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. p. 1715–1721. GECCO '19, ACM (2019)
15. Petke, J., Haraldsson, S.O., Harman, M., Langdon, W.B., White, D.R., Woodward, J.R.: Genetic improvement of software: A comprehensive survey. IEEE Trans. Evol. Comput. **22**(3), 415–432 (Jun 2018)
16. Sarti, S., Adair, J., Ochoa, G.: Recombination and novelty in neuroevolution: A visual analysis. SN Computer Science **3**(3) (Mar 2022)
17. Trefethen, L.N., Bau, D.: Numerical Linear Algebra. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1997)