Reaching Meaningful Diversity with Speciation-Novelty in Genetic Improvement for Software

Zsolt Németh
School of Computing and Communications, Lancaster University
UK
z.nemeth@lancaster.ac.uk

Penn Faulkner Rainford
Department of Computer Science, University of York
UK
penn.rainford@york.ac.uk

Barry Porter
School of Computing and Communications, Lancaster University
UK
b.f.porter@lancaster.ac.uk

Abstract

Genetic Improvement (GI) for software has been used in automated bug fixing and in automated performance improvement. Automated improvement has been targeted at multi-context problems, where one implementation variant might be best at one context, and another might be best at a different context. However, this application of GI generally requires a fresh improvement process for each new context, which can be computationally expensive. We propose a novel application of GI for multi-context problems, in which we aim for a diverse set of individuals in an initial training run for one context. We use a phenotypic speciation metric as a diversity indicator, allowing us to plot a diversity geometry through program search space. When a different context is introduced, as a new optimisation target for GI, we are able to select from one of these diverse individuals as a close starting point for fine-tuning. With a hash table implementation as an example to genetically improve, we show that we can exercise a high degree of control over population diversity, and that this diversity can be a useful starting point for finding individuals in successive alternative contexts.

Keywords

Genetic programming, Genetic improvement, Genetic diversity, Speciation, Novelty search, Fitness landscape

1 Introduction

Genetic Improvement (GI) for software has been used extensively in automated bug fixing [22, 23], and more recently in automated performance improvement [26]. Automated improvement itself has recently been targeted at multi-context problems, where one implementation variant might be best at one context, and another might be best at a different context [27]. However, this application of GI currently requires a fresh improvement process for each new context, which can be computationally expensive.

We propose a novel application of GI for multi-context problems, in which we deliberately aim for a diverse set of individuals in an initial training run for one context. When a different context is introduced, as a new optimisation target for GI, we are then able to select from among these diverse individuals as a close starting point for fine-tuning. We employ the recently-proposed phenotypic speciation theory [1] as our diversity indicator, and target our GI process at a hash table implementation which receives different collections of keys to store, representing different contexts.

Our approach has similarities with novelty search [6], in which a search process attempts to avoid local minima by quantifying the value of individuals by their novelty; in contrast to this, we use a numerical phenotypic species classifier as a proxy for overall search space geometry, allowing us to seek particular diversity distributions across this geometry as our diversity target.

Our specific research questions are:

- To what extent can we control the diversity level of individuals that a GI process yields?
- How different is a population of individuals that have been deliberately sought for diversity, to a population of highly-optimised but different individuals?
- What is the utility of diverse individuals in subsequent GI processes for new contexts?

Our results demonstrate firstly that we are able to exercise a surprisingly high degree of control over diversity, allowing us to change the population's collective genetic diversity in specific ways while remaining within the envelope of correct solutions to a given problem. Second, we show that a species-oriented diversity approach is able to find more meaningfully diverse individuals than a mixed starting population of highly-tuned individuals. And third, we show that in new contexts we are able to specialise from a diverse set of individuals more quickly than a fresh execution is able to, demonstrating the value of deliberately forming populations of meaningful diversity. In future work we aim to apply these methods to additional problem domains.

In the remainder of this paper we discuss closely related work in Sec. 2, then present our approach to species-driven diversity search in Sec. 3. We present our empirical evaluation in Sec. 4 and conclude in Sec. 5.

2 Related Work

Our research touches on the topics of fitness landscape navigation; speciation in GI; and alternative approaches to GI operating specifically on hash functions. We present closely related work across each of these areas.

The fitness landscape. The concept of fitness landscape was coined by Wright [33] as a terrain map with peaks, valleys, and plains. Wright's idea is widespread in evolutionary computation scenarios, e.g. [21, 34]. Petke et al. [22] review case studies of GI fitness landscapes in particular and establishes some of their properties. To a large extent, the shape of the program search space landscapes are still unknown; furthermore, due to the nature of the case studies (where each work tends to focus on specific features), these works reveal facets of the landscape but not a coherent and general view.

In GI, navigating neutral drift and understanding epistasis have become of prime interest in better covering large search spaces.

Bruce et al. [4] investigate the frequency of meaningful mutations in the search space, but since their fitness function is energy consumption, this again yields a generally continuous landscape composed of plains and gradients. The work by Langdon et al. [17] focusses on the robustness of code to mutation, and the effects of single mutation steps, which corresponds to local analysis of a singular point and its neighbours and provides indirect information about the full extent of the search space. Haraldsson et al. [13] operate on a dual landscape, considering both functional (number of test cases passed), and non-functional (execution time). The landscape is investigated by random walks of 10 steps and first-order mutants of a limited set of operations, with similar conclusions to [17]. Van Laar [30] presents a systematic analysis of the search space by random walks and tries to explore the extent of the plateaus. The experiment is in a code improvement for correctness scenario, hence changes in fitness are very rare, and both random walks and search for the edge of a plateau are possible for an extreme number of steps in a neutral space.

In the context of the above work, this paper demonstrates a novel way to examine fitness landscapes, though the lens of speciation diversity, and its effect on convergence towards particular solutions.

Species. The seminal paper by Goldberg et al. [11] introduced *niching by fitness sharing*, that conceptually corresponds to species. Their method requires a distance definition (either in the genotype or phenotype space), and a sharing function; fitness values of each individual are adjusted so that individuals close to each other share their fitness values (and the proportion of sharing decreases by distance). Thus, the population is subdivided and competition between distant points in the search space is controlled; and as a side effect, the diversity was also increased. The explicit definition of species was first introduced by Cioppa et al. [5] which builds on the above work. Goldberg et al. [12] further refined their initial concept by an adaptive niching method. Li et al. [19] took this further, with evolving parallel sub-populations where species are defined by Euclidean distance in their genotypes capturing the similarity of individuals, and a constant delimiter to separate species; species in this model are therefore clusters of individuals around the best individual. This approach reveals and addresses the otherwise contradicting elitism (conserve the fittest) and diversity (conserve the different) that is in focus of our work. [20] presented an advanced, adaptive version of species conservation. Dong et al. [7] present a variation of [19] that after a seeding procedure the produced clusters remain static and all further genetic interactions are intra-species.

Jelasity et al. [15] built on similar assumptions, i.e. Euclidean distance between individuals, seeds and the species distance threshold are defined but their species delimiter is a monotonic decreasing function that controls the ability to escape from a local optima. Species act within their radius of attraction on disjoint subdomains allowing to explore different local optima. The model by Raghuwanshi et al. [25] divides the population into males and females and species are determined by niching around the female individuals based on Euclidean distance. Similarly to [15], interaction is possible within the species and merging of species is also allowed if they converge. Wong et al. [32] try to improve species conservation [19] not only by keeping species alive, but ensuring a sufficient number of individuals by a species-specific explosion, so that evolution and convergence are possible in each species.

As far as we are aware, our use of phenotypic speciation to gain a search space geometry for diversity-focused search is entirely novel.

Evolutionary hash functions. The creation of a hash function by genetic programming has been addressed by Berarducci et al., Hussain et al. [3, 14]. Several authors tried to improve hashing using genetic programming, such as Estebanez et al. [8, 9] to improve the nonlinearity (avalanche effect) of hashing and Safdari et al. [29] to minimise collisions. The goal set by Saez et al.[28], applying evolutionary techniques for designing "ad hoc" hash functions adapted to real-world dynamic environments is close to ours. All of these works, however, approach the problem from a genetic programming aspect, i.e., creating hash functions by evolution as opposed to our genetic improvement at the source code level.

3 Method

Our research addresses the quest for algorithmic diversity by speciation, transforming the fitness landscape and applying novelty search. We use speciation as our guide to differentiate between diversity at a tokenistic or source-code level and diversity that is *meaningful*: that which results in a measurably different behaviour. In this section we first introduce our base implementation of genetic improvement for source code, then present our approach to defining, quantifying and controlling meaningful diversity.

3.1 Base GI Implementation

Our GI system uses abstract syntax tree representations of source code, applying successive rounds of both mutation and crossover to individuals. The multi-context nature of our research stems from the emergent software systems field [10] in which a system is likely to be subjected to multiple discrete operating environments over the course of its running time, caused for example by changes in request patterns, changes to network characteristics, or fluctuations in available energy. The emergent software systems field examines this effect at a macro-level of software architecture, selecting different architectural variants (and sub-architectures) which best suit the current deployment context.

We examine this area at a micro-level of individual software components, in which we can produce an implementation variant of a given component which is better tuned to the current deployment context. The utility of diversity at this level is evidenced by the wealth of different search, sorting, scheduling, and caching algorithms available, to name a few.

We assume that a particular sub-component in a running system is selected as a candidate for improvement, at which point a function call trace monitor is temporarily injected into the system at the interface to this component. This monitor logs every function call into the component of interest, including its parameter values and return values. This produces a trace of function calls which represents the local symptoms of the overall operating context of the wider system. We can then re-play this trace of function calls to mutated variants of this component during a GI process, to attempt to yield an improved variant of the component for this operating

context. We specifically choose a hash table component as our target for improvement, which aligns our research with other work in this area (e.g. [26]).

Our GI framework therefore parses the source code of a given hash table implementation, and is instructed to focus on the hash function which controls the distribution of keys among buckets. The interaction between the hash function logic, and the set of keys it is being presented with, yields the resulting distribution: the ideal distribution is generally thought to be a uniform one, such that for 1,000 keys, with 100 available buckets, the hash function will cause 10 keys to be placed in each bucket, therefore minimising the average key retrieval speed. Highly disuniform access patterns of keys may skew this trait, however, such that highly-accessed keys benefit from being in buckets with fewer items.

Our GI process uses mutations that can insert newly synthesised lines of code (such as new variable declarations, new assignments, or new control-flow constructs such as if-statements); can delete existing lines of code; and can mutate existing lines of code by changing operators or operands. The set of available mutation operations for a chosen mutation point is filtered to those that are most likely to remain semantically valid, reducing the number of non-compilable individuals.

Crossover is realised as horizontal (or lateral) gene transfer where genetic material moves between organisms that are not in parent-offspring relationship. In this case, parts of the source code, represented by a token of the abstract syntax tree, are inserted into the code text (genotype) of another individual. The probability of mutations and crossovers, the target token and the type of the genetic operations are all selected randomly, using weighted roulette wheel methods and the associated weights. For the purpose of our research in this paper, all weights are configured as uniformly as possible (with the caveat that some mutations are inherently more semantically viable than others in particular code formulations).

Fitness is evaluated as simple runtime against a function call trace, such that lower runtime values mean better fitness. Finally, in the selection phase, the population is divided into those individuals, that would have offsprings for the next generation, and those that would be extinct. In the base model, individuals are ranked by their fitness and then, chosen randomly, weighted by their rank and performance.

3.2 Quantifying Diversity

GI for software is profoundly different from other GI optimisation methods. The genotype is the source code itself, whereas its runtime characteristics, such as what algorithm it realises, how fast it is, and how much resources it needs form attributes of its phenotype. Although significantly different source codes (genotypes) can represent the exact same algorithm (phenotypes), minute changes in the source code can bring dramatic changes in runtime behaviour. Specifically, in the realm of GI for software improvement, diversity in genotype does not imply diversity in phenotype (nor does similarity in genotype imply similarity in phenotype).

To capture meaningful behavioural diversity among source code candidates, we adopt a recently-proposed phenotypic speciation classification mechanism, which captures how an implementation behaves rather than the fine details of how that implementation is encoded [1]. The grouping of individuals by their *ecological niche*, closely corresponds to the concept of speciation, and therefore our conception of classification somewhat resembles the notion of *ecospecies* [31]. Our *definition of species* divides the phenotype of an individual into its **functional** (the algorithm it represents) and **non-functional** (how the algorithm is represented) attributes. Individuals are classified according to their **functional** phenotype in such a way that the definition is *selective* (able to detect the smallest algorithmic changes), *characteristic* (a fingerprint-like identification of individuals of a species), and *insensitive* to implementation details while also being easily computable. Thus, individuals are *meaningfully different* if they realise different algorithms and not just variations of the same algorithm.

Particularly, in the case of a hash table of n buckets and a hash function $h : \mathcal{S} \mapsto \{1 \dots n\}$, the layout of the table at the end of the training run, i.e., how many elements each bucket holds, is characteristic of the hash algorithm. Therefore, the layout is captured as a vector of relative frequencies of each element

$$P(x) = \frac{\text{elements in bucket } x}{\text{all elements stored in the hash table}}. \tag{1}$$

These vectors are comparable by *probability divergence metrics*, and the algorithms are categorised by their *distance from the uniform distribution*. (In a strict sense this categorisation cannot identify the algorithm to the smallest details, such as it is insensitive to different permutations of the buckets. However, from a behavioural point of view, these hash functions are identical.) We applied the Kullback-Leibler [16] divergence metrics, and the distance of $P(h(s) = x)$ over $\{1 \dots n\}$ from the uniform distribution Q over $\{1 \dots n\}$, $Q(x) = \frac{1}{n}$ is established as:

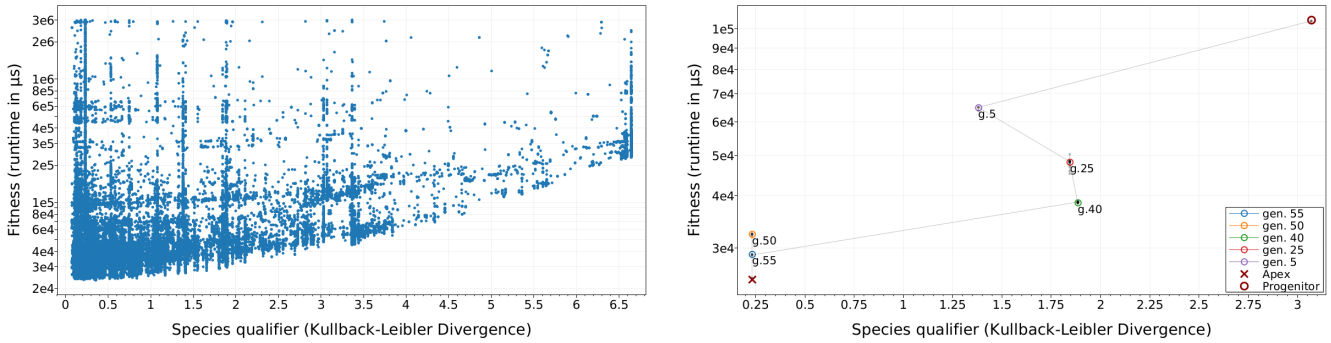
$$D_{KL}(P \parallel Q) = \log n + \sum_{x \in \{1 \dots n\}} P(x) \log P(x) \tag{2}$$

We *define* species by one particular functional attribute of the phenotype: the calculated Kullback-Leibler divergence. Two individuals are therefore of the same species, and therefore not meaningfully diverse, if their D_{KL} values are identical, regardless of their genotype and other attributes of the phenotype.

3.3 Transforming the fitness landscape

The use of phenotypic species also enables us to redefine the search space as a geometry of diversity. Conventionally, the fitness landscape is imagined in a space spanned by genotype-phenotype coordinates, but considering the lack of correlation between the source text and the runtime behaviour explained in Section 3.2, the landscape would be extremely rugged or even meaningless for the GI of the source code. The Kullback-Leibler divergence metric determines the possible range of quantified species within known bounds: from the uniform distribution $D_{KL} = 0$ to the most uneven distribution, where all elements of the training set are placed in the same bucket and leave everything else empty, $D_{KL} = \log n$, which is 6.64 for a hash table of 100 buckets. Since we know the lower and upper bounds of the numerical range of species, and the fitness values obviously have these bounds, we can use these ranges as a proxy to represent the total geometry of the search space.

Figure 1a shows the extent and shape of program search space when projected using phenotypic speciation, as approximated by



(a) Geometry of species-fitness space as approximated by individuals. **(b) One evolutionary trajectory in the species-fitness landscape.**

Figure 1: The species-fitness landscape

2678493 individuals of 965 GI experiments, using a particular trace of function calls. Here, the species quantifier of individuals is shown on the x-axis, and the fitness of the individual is plotted on the y-axis. Vertical lines in this figure represent different versions of the same algorithm with varying fitness values. Figure 1b depicts the trajectory of one particular apex individual as it traverses the landscape through generations. Meaningful, algorithmic changes, e.g., between generations 0 and 5, appear as shifts on the species (x) axis, whereas oscillations along the fitness (y) axis, e.g. at generation 25, represent tuning of the same algorithm. This alternative view of the fitness landscape and its geometry make possible to conceptualise algorithmic diversity as a measure of the *spread of individuals across the x-axis* in the search space, and reason about evolution in geometric terms.

3.4 Search for Diversity

Selection is a key aspect of all genetic methods: a given strategy is used to eliminate the least fit individuals, making space for the more fit ones. This is often combined with a degree of elitism, protecting the *very best n* individuals from mutation or incoming gene transfer. Selection generally ranks individuals by their fitness, with the goal to increase the overall fitness of the population. We refer to this as a **fitness-first** strategy.

In contrast to this, we introduce a **species-first** selection strategy to overcome the contradiction between elitism and diversity [19]. It prioritises individuals of new species (those that realise a *different algorithm*), irrespective of their actual fitness. This somewhat corresponds to the concept of novelty search by [18], that instead of rewarding performance on an objective, rewards diverging from prior behaviours. Our phenotypic speciation captures behavioural difference in a measurable and quantitative way, such that the species-fitness landscape corresponds to the theoretical behaviour space. Hence, the proposed species-first selection strategy is a conceptual variation of the novelty algorithm, but with an entirely different implementation both in measurement of difference and in selection approach. One can consider the species-first approach as a breadth first approach as it tries to explore many variants spanning a significant range of the species spectrum, that is necessarily followed by a fitness-first (depth-first) search where the search focuses on a significantly narrower range of species and driven by fitness. In our approach the species-first selection keeps the 2 best individuals of each newly spawned species, while the rest of the population undergoes further mutations and crossovers. In a population of 50 individuals, 25 pairs of different species can thus be found by the breadth first method.

All breadth first searches produce a large number of individuals with poorer fitness, therefore, at some point a switch in selection strategy is necessary to optimise towards a specific target. To quickly eliminate the unpromising individuals and go depth-first selection at the hot spots for optimum, we introduced a more aggressive version of elitism where not only the best individuals are protected but also the bottom portion of the population is given no chance to survive.

4 Evaluation

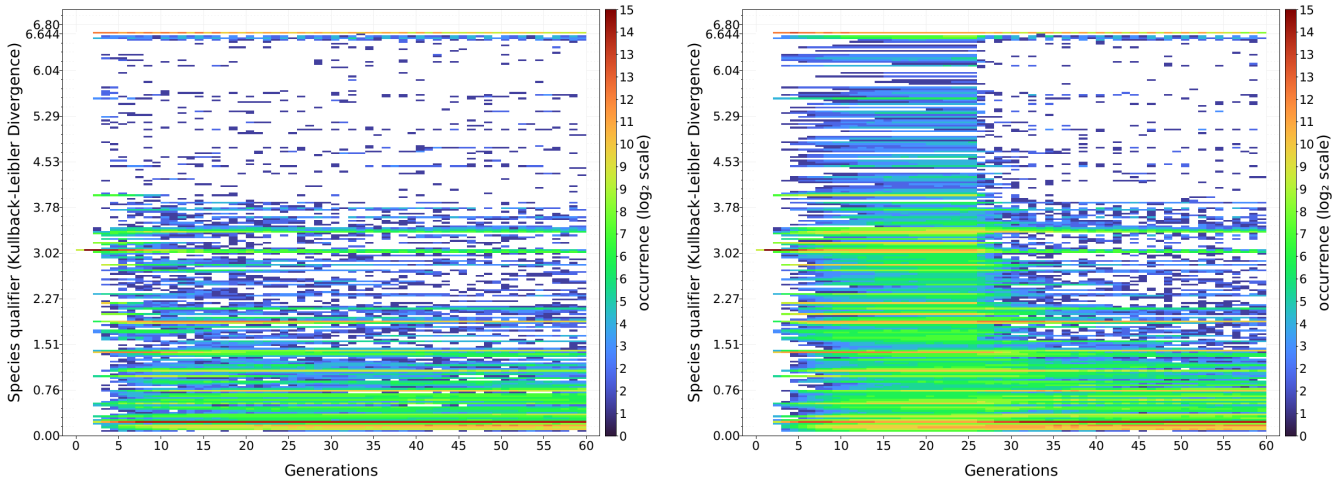
4.1 Experimental setup

Our experiment conditions are designed around multiple different function call traces, representing different contexts for a hash table. Our call traces include those that use key values that are English words, those that use Polish words, those that use numeric values for keys, and a set of synthetic function call traces designed to exercise particular edge cases.

Each experiment uses populations consisting of 50 individuals and evolution runs for 60 generations. The probability of mutation is 0.8; weights of insert, delete, and modify mutations are 1/3-1/3, static; the probability of crossover is 0.2. Elite is composed of 2 individuals, selection weights are calculated by a reciprocal method. The experiments for side-by-side comparison were repeated 480 times and the results are collated. Our GI framework is made available for replication [2].

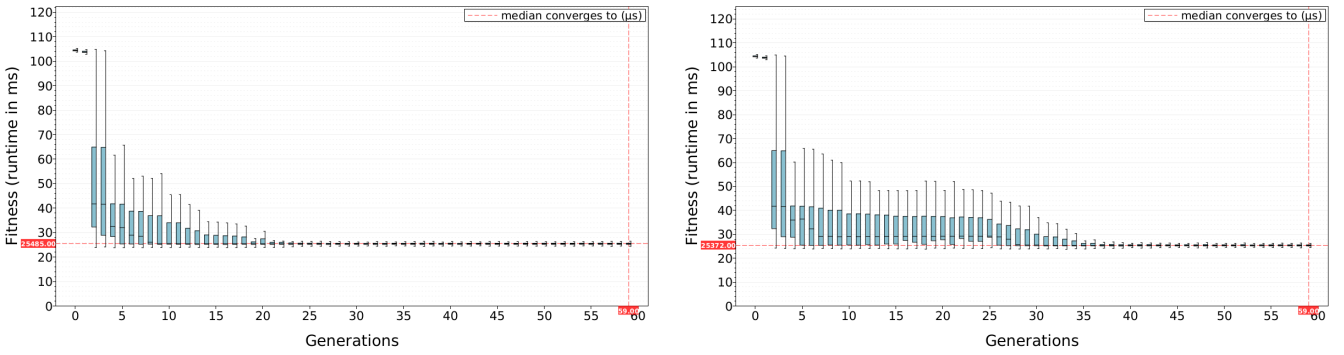
Using this setup, we seek to answer the following three research questions, as outlined in Sec. 1:

(1) To what extent can we control the diversity level of individuals that a GI process yields?
(2) How different is a population of individuals that have been deliberately sought for diversity, to a population of highly-optimised but different individuals?
(3) What is the utility of diverse individuals in subsequent GI processes for new contexts?



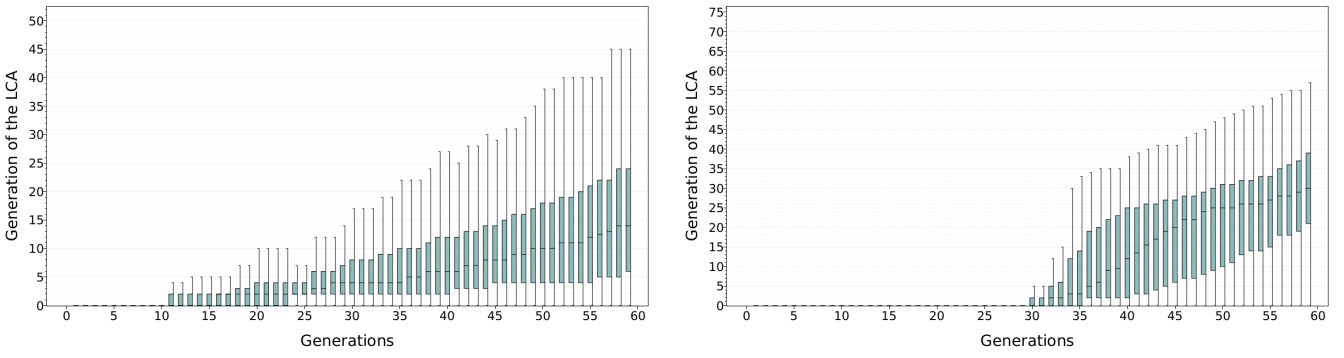
(a) Fitness-first selection strategy. **(b) 25 generations of species-first, then fitness-first selection strategy.**

Figure 2: Histogram of species distribution over time, multiplicative hash, English training set.



(a) Fitness-first selection strategy. **(b) 25 generations of species-first, then fitness-first selection strategy.**

Figure 3: Distribution of the best fitness values over time, multiplicative hash, English training set.



(a) Fitness-first selection strategy. **(b) 25 generations of species-first selection strategy, then fitness-first.**

Figure 4: Distribution of the generation of Lowest Common Ancestors (LCAs)

For question (1), we examine the level of meaningful diversity that we gain from a species-first selection strategy compared to a fitness-first strategy. For question (2), we examine the resulting diversity level compared to an experiment which starts from mixed populations of highly-tuned individuals. For question (3), we examine the effect of switching between different contexts after an initial species-first period.

4.2 Results

Question 1. We first conduct a baseline experiment where the training set for our function call trace is 1,000 English words used as keys. We first use a fitness-first selection strategy by itself, then compare this with a species-first selection strategy up to generation 25 followed by a fitness-first selection strategy. To answer research question (1) we observed the overall species diversity and its effect on non-functional attributes.

Figure 2 compares the temporal distribution of species for the two experiments across generations. The graphs are histograms represented as heat-maps: they present the occurrence counts of species in a given range, quantised into 220 bins, i.e. each bin spans 0.03 of the entire species range of 0 . . . 6.64. Quantities are mapped to colours on a logarithmic scale to emphasise the minor differences at the lower end, thus giving a quick qualitative evaluation.

In Figure 2a we see a starting point of a single individual at species $D_{KL} = 3.07$. After a brief surge of this species, it becomes marginalised, while other species in the 0 . . . 1 range dominate, most typically around ~0.1 . . . 0.2. From one starting point, the mass of species gradually moves to a range covering a lower fifth of all possible species. This is in alignment with the landscape geometry in Figure 1 that suggests that better runtimes for this training set are to be found at species in the lower region. Abrupt fluctuations in the mass of species, i.e. thin green/yellow lines among darker green/blue stripes illustrate the nature of genetic evolution: some species are more easy to reach by mutation and crossover than others. We note that the species with $D_{KL} = 6.64$ is one of the most easily reachable species, as it represents the case where all elements of the training set are put into the same bucket. This can happen in many ways, making it a common occurrence: by inserting an assignment of a constant; by deleting the core of the inner loop; by altering an operation so that it results in constant values, etc. While individuals of this species are functionally correct, they are tremendously inefficient, with little chance to survive.

Figure 2b shows the same experiment where we instead use our species-first selection strategy for 25 generations to attempt to maximise meaningful diversity. As can be seen, until g. 25, the cover of the species dimension improves significantly, giving a good probability (on average) of having individuals in the ~0 . . . 3.5 range. For the upper half, the presence of individuals is less prominent, yet the overall cover of this range is much denser than in the fitness-first experiment. In detail, over 90% of histogram bins contain at least 1 element, with 65% containing at least 10, compared with at most 50% of bins are non-empty and 20% having more than 10 elements in the fitness-first result. After g. 26, when we switch to a fitness-first selection approach, the mass of the individuals shifts toward the spot of the optimum in the ~0.1 . . . 0.2 area.

Next, we check how our controlled diversity can be traced in other metrics. Figure 3 compares the distribution of fitness for the fittest individuals in successive generations, the lower and upper half of the boxes represent Q2 and Q3, and the line between is the median. The experiment in Figure 3a starts with a single individual of fitness ~105000μs. The median fitness of the best individuals drops shortly to ~30000μs while the overall range of the best fitness values increases from ~22000μs to ~50000μs. Then this range narrows as evolution improves the running times (note that *all* individuals are the best of their generation, hence they cannot be extinct by selection.) Eventually, the best fitness stabilises at ~25000μs around g. 20.

In Figure 3b fitness initially drops to ~40000μs as in Figure 3a but then the convergence stalls: the selection prioritises new species instead of better fitness. At generation 25 the selection switches to the fitness-first strategy and the best fitness is stabilised at the same level as in the previous experiment. Thus, there is no difference in the overall improvement in fitness. However note, that the convergence speed is different, such that the more diverse starting point yields a somewhat shorter, 10 generation convergence time.

Figure 4 compares the effect of selection strategies on the Lowest Common Ancestors (LCAs) that characterises specialisation. The LCA for a group of individuals g is another individual $LCA(g)$ higher up in the ancestry tree, so that all members of the group g are descendants of the LCA and no other common ancestors can be found among the descendants of LCA. Specialisation in this context means a common genetic history of a group of individuals from the root of the ancestry tree to the LCA. LCA is also a measure of similarity in genotypes; LCAs earlier in the ancestry tree suggest that individuals in g are more dissimilar (their common genetic history is short), while later LCAs mean that the genetic split in the phylogenetic history is more recent, and thus the individuals of g are more similar (they have more common genetic history). [27] argue that highly specialised individuals, that have a longer evolutionary history, are less capable to adapt to changing environments than those higher up in the ancestry tree.

In Figure 4 we present the statistical distribution of the generation of LCAs of an entire generation. For example, the box in Figure 4a, g. 59 shows that the LCA for individuals in the last generation is in or higher than g. 14 in half of the cases; g. 10 shows that LCA for this generation is in g. 0 in all cases, there is no specialisation at all until g. 11; in other words, the population of these individuals have at least two distinct evolutionary paths spawned at the root.

It is clearly visible in Figure 4b that the species-first selection strategy defers the start of specialisation, such that all generations until g. 30 have their LCAs in g. 0. This validates our expectations, as the search for novel algorithms suggests substantial differences in their implementation, hence their genotypes are expected to be more different indicated by the highest possible LCA (the root). In other words, at the moment of context change in the environment, i.e., g. 25, the population was measurably more diverse than in case of an fitness-first strategy. Also, the median generation of the LCA for the apex generation (g. 59) is g. 14 (and in half of the cases the LCA is between g. 6 and g. 24) for the fitness-first strategy in Figure 4a; and g. 30 (in half of the experiments the LCA is between g. 20 and g. 39) for the species-first strategy in Figure 4b.

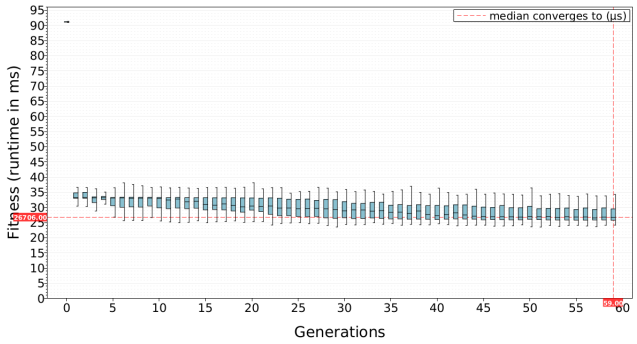


Figure 5: Distribution of the best fitness values over time with fitness-first selection strategy, started from a population composed of 6 different hash algorithms.

This suggests that individuals of the apex generation are better specialised (have longer common genetic history) in case of the species-first selection. Increased diversity and better specialisation suggests a more certain path close to the optimum.

The first round of experiments answer question (1) and confirmed that (i) it is possible to control algorithmic diversity by speciation and species-first strategies (ii) the controlled diversity results in a better coverage of the species dimension, and (iii) it has positive effects on the convergence time and on the overall specialisation.

Question 2. Next, to answer question (2) we compare our results to an experiment, where six different, well-known hash algorithms form the initial generation (such as Kernighan-Ritchie, Bernstein, Fowler-Noll-Vo, etc.). Figure 5 illustrates the distribution of the best running times of collated experiments for this condition, showing that running times stay within a tight group despite the apparently diverse starting set. We compare this with two of our earlier conditions in which we start from a population formed of identical individuals of a single hash function implementation: one condition in which we use only a fitness-first configuration, and the other in which we start from species-first and transition to fitness-first.

In the single-individual fitness-first condition, shown in Figure 3a, we see a much broader range of fitness values for the first few generations, followed by a sharp reduction in this range as the GI process specialises its individuals. In the mixed population case, albeit there is improvement over time, some of the individuals are already close to the optimum, whereas in case of our single algorithm fitness-first condition quite a few generations are necessary to reach this specialised stage.

Our single-individual species-first condition, shown in Figure 3b, maintains a broad fitness range for the whole of its 25 generations in the species-first configuration. Following this point, from a set of diverse individuals, it reaches a specialised population very quickly.

This comparison answers question (2), and confirms that from a single individual the species-first strategy can create a meaningfully diverse population similarly to an inherently mixed starting population.

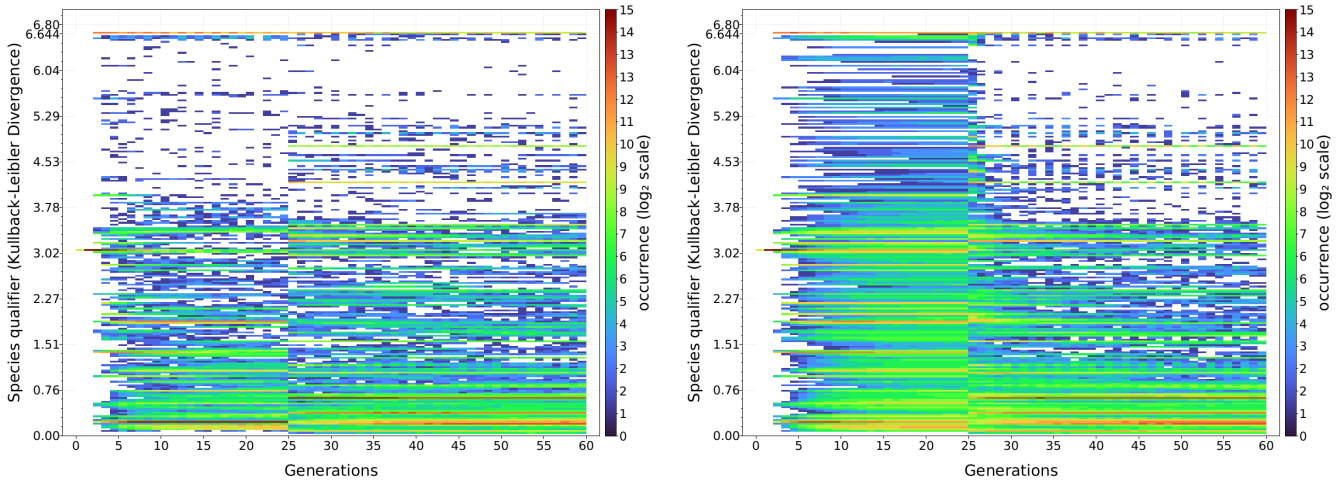
Question 3. The same experiments were repeated, this time with a simulated *context switch* in order to see the effect of increased diversity in a changing environment. Experiments were started with the same training set of 1,000-word English keys used earlier, subsequently switching to a numeric data set of 1,000 EEG samples at g. 25. As before, one set of experiments uses a fitness-first selection strategy from the start, while the other uses species-first strategy up to g. 25 and then fitness-first.

Figures 6a and 6b compare the effect of selection strategies on the diversity. As in the previous case, the species-first strategy gives a significantly better coverage of the species range, over 90% vs 52% for non-empty bins and 65% vs 22% for bins with at least 10 elements. The convergence times are also improved from ~22 generations of pure numeric training to ~3 generations when the population was pre-trained with English text and species-first strategy the context switch in g.25 (graph not shown.) The prolonged stall of specialisation and the eventual stronger specialisation are also present in the LCA data of the pre-trained population (graph not shown, qualitatively similar to Figure 4.)

The speciation-based diversity search enables an effective control of meaningful diversity, and increased diversity plausibly improves the ability to navigate the fitness landscape even if context switches occur. The convergence time is improved by pre-training, compared to a population without pre-training. However, there is no measurable effect of selection strategies on the convergence times if the evolutions of two pre-trained populations are compared. Figure 7 shows the distribution of best runtimes in each generation, and their convergence to the optimum with fitness-first (Figure 7a) and species-first (Figure7b) selection, respectively. There is no difference in the optimum, and the convergence time in case of species-first selection is slightly worse.

The reason we do not see an improvement in this metric is in the simplistic nature of a hash function and its fitness landscape: instead of crisp point of optimum, it has a large area of nearly equally-good fitness across contexts. Hence, reaching a quasi-optimum (close to the optimum) point with one training set also results in a quasi-optimum point for another training set, irrespective of the selection strategy. For example, in Figure 6b the most frequent species just before the context switch in g. 25 are ~0.2 . . . 0.3, ~0.6, ~1, ~1.4, ~1.9, ~3.1, ~3.3, whereas the most frequent species at the end of the numeric training in g. 59 are ~0.25 . . . 0.35, 0.4, 0.6. In case of fitness-first selection in Figure 6a, the most frequent species at the end of g. 25 are in 0.2 . . . 0.3: the frequent species found by the fitness-first strategy are a subset of the results of the species-first selection. Since the eventual optimum range is close to ~0.2 . . . 03, and it was found by both fitness-first and species-first selection there is little difference in convergence times.

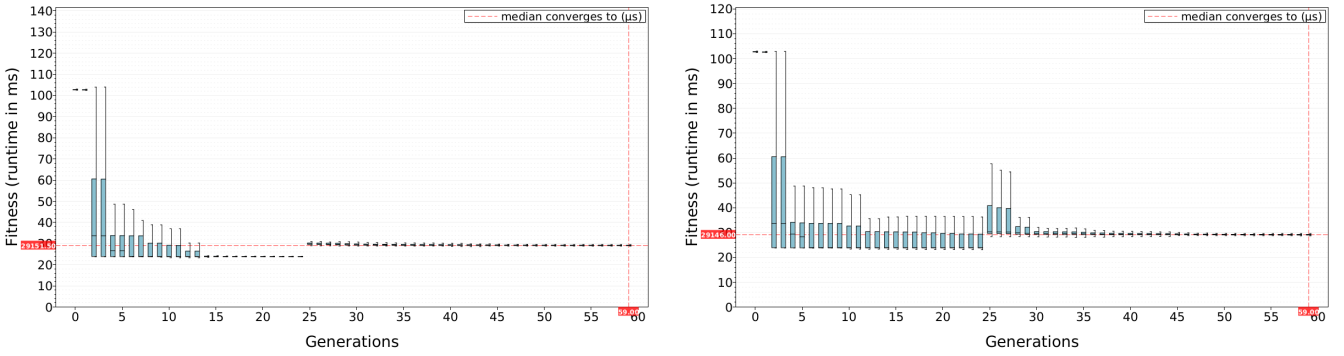
The answer to question (3) is therefore mixed. We can clearly observe the increased diversity, shorter convergence time, and postponed specialisation; in this aspect, the answer is positive. However, we cannot prove that the shorter convergence time is a consequence of the species-first selection, as similar convergence times were recorded in experiments with entirely fitness-first selection. In future work we will examine more complex application domains to revisit this question in particular.



(a) Fitness-first selection strategy all along.

(b) Species-first selection until g. 25, then switch to fitness-first.

Figure 6: Histogram of the species. Context switch in g. 25, the English training set replaced to a numeric one.



(a) Fitness-first selection strategy.

(b) Species-first selection until g. 25, then switch to fitness-first.

Figure 7: Distribution of the best fitness values over time. Context switch at g. 25, the English training set replaced to a numeric one.

5 Conclusion

Software systems are often exposed to diverse deployment environment conditions across their operational lifetime; in these scenarios it tends to be the case that different sub-component implementations are better suited to those different conditions [24]. In this paper we have used GI to derive new implementation variants which are better optimised for particular conditions.

Diversity is a key metric in such scenarios, either when the starting point is a single code variant (single genotype) and a population of various capabilities must be built up in a controlled way; or when distant points of the fitness landscape must be explored simultaneously; or in a changing environment, a non-homogeneous population has better chance to contain individuals that are best suited to adapt to the new conditions. In particular, we address the diversity aspect of genetic improvement: whether diversity can be quantitatively captured and controlled. This challenge is non-trivial in GI for source code, as meaningful, algorithmic diversity cannot be established from static code analysis; rather, behaviour analysis is necessary at runtime.

We established a method constituted of a phenotypic speciation that captures and characterises algorithmic differences independently from the source code; a transformation of the potentially infinite, extremely rugged search space into a bounded and geometrically conceivable fitness-species landscape; and a species-first selection strategy that counteracts the effect of elitism and prioritises behavioural difference to optimise to a specific objective. Empirical evaluation has shown that our approach is able to control the meaningful (phenotypic, algorithmic) diversity and thus, control convergence time and specialisation. It was shown that our method can create diversity comparable to inherently mixed populations. The effect of increased diversity on the overall evolutionary process is demonstrated by lower convergence times compared to a GI run which had started from a single individual; in future we will further validate our work in other scenarios.

Acknowledgments

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

References

[1] Anonymous. 2024. Undisclosed Title *(Artificial Life Conference Proceedings, Vol. ALIFE 2024: Proceedings of the 2024 Artificial Life Conference).*

[2] Anonymous. 2025. Addendum to "Reaching Meaningful Diversity with Speciation-Novelty in Genetic Improvement for Software". https://doi.org/10.5281/zenodo.14771561

[3] Patrick Berarducci, Demetrius Jordan, David Martin, and Jennifer Seitzer. 2004. GEVOSH: Using Grammatical Evolution to Generate Hashing Functions. In *GECCO 2004 Workshop Proceedings*, R. Poli, S. Cagnoni, M. Keijzer, E. Costa, F. Pereira, G. Raidl, S. C. Upton, D. Goldberg, H. Lipson, E. de Jong, J. Koza, H. Suzuki, H. Sawai, I. Parmee, M. Pelikan, K. Sastry, D. Thierens, W. Stolzmann, P. L. Lanzi, S. W. Wilson, M. O'Neill, C. Ryan, T. Yu, J. F. Miller, I. Garibay, G. Holifield, A. S. Wu, T. Riopka, M. M. Meysenburg, A. W. Wright, N. Richter, J. H. Moore, M. D. Ritchie, L. Davis, R. Roy, and M. Jakiela (Eds.). Seattle, Washington, USA. http://gpbib.cs.ucl.ac.uk/gecco2004/WUGW001.pdf

[4] Bobby R Bruce, Justyna Petke, Mark Harman, and Earl T Barr. 2019. Approximate oracles and synergy in software energy search spaces. *IEEE Transactions on Software Engineering* 45, 11 (2019), 1150–1169.

[5] Antonio Della Cioppa, Claudio De Stefano, and Angelo Marcelli. 2007. Where Are the Niches? Dynamic Fitness Sharing. *IEEE Transactions on Evolutionary Computation* 11, 4 (2007), 453–465. https://doi.org/10.1109/TEVC.2006.882433

[6] Stephane Doncieux, Alban Laflaquière, and Alexandre Coninx. 2019. Novelty search: a theoretical perspective. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 99–106.

[7] Na Dong, Chun-Ho Wu, Wai-Hung Ip, Zeng-Qiang Chen, Ching-Yuen Chan, and Kai-Leung Yung. 2011. An improved species based genetic algorithm and its application in multiple template matching for embroidered pattern inspection. *Expert Systems with Applications* 38, 12 (2011), 15172–15182. https://doi.org/10.1016/j.eswa.2011.05.085

[8] César Estébanez, Julio César Hernández-Castro, Arturo Ribagorda, and Pedro Isasi. 2006. Evolving Hash Functions by Means of Genetic Programming. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (Seattle, Washington, USA) *(GECCO '06)*. Association for Computing Machinery, New York, NY, USA, 1861–1862. https://doi.org/10.1145/1143997.1144300

[9] César Estébanez, Yago Saez, Gustavo Recio, and Pedro Isasi. 2014. AUTOMATIC DESIGN OF NONCRYPTOGRAPHIC HASH FUNCTIONS USING GENETIC PROGRAMMING. *Computational Intelligence* 30, 4 (2014), 798–831. https://doi.org/10.1111/coin.12033 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/coin.12033

[10] Roberto Rodrigues Filho and Barry Porter. 2017. Defining Emergent Software Using Continuous Self-Assembly, Perception, and Learning. *ACM Trans. Auton. Adapt. Syst.* 12, 3, Article 16 (Sept. 2017), 25 pages. https://doi.org/10.1145/3092691

[11] David E. Goldberg and Jon T. Richardson. 1987. Genetic Algorithms with Sharing for Multimodalfunction Optimization. In *Proceedings of the 2nd International Conference on Genetic Algorithms, Cambridge, MA, USA, July 1987*, John J. Grefenstette (Ed.). Lawrence Erlbaum Associates, 41–49.

[12] D. E. Goldberg and L. Wang. 1997. Adaptive niching via coevolutionary sharing. (1997), 21–38.

[13] Saemundur O. Haraldsson, John R. Woodward, Alexander E. I. Brownlee, Albert V. Smith, and Vilmundur Gudnason. 2017. Genetic improvement of runtime and its fitness landscape in a bioinformatics application. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Berlin, Germany) *(GECCO '17)*. Association for Computing Machinery, New York, NY, USA, 1521–1528. https://doi.org/10.1145/3067695.3082526

[14] Daniar Hussain and Steven Malliaris. 2000. Evolutionary Techniques Applied to Hashing: An efficient data retrieval method. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, Darrell Whitley, David Goldberg, Erick Cantu-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer (Eds.). Morgan Kaufmann, Las Vegas, Nevada, USA, 760. http://gpbib.cs.ucl.ac.uk/gecco2000/RW054.pdf

[15] Márk Jelasity and József Dombi. 1998. GAS, a concept on modeling species in genetic algorithms. *Artificial Intelligence* 99, 1 (1998), 1–19. https://doi.org/10.1016/S0004-3702(97)00071-4

[16] S. Kullback and R. A. Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22, 1 (1951), 79–86. https://doi.org/10.1214/aoms/1177729694

[17] William B. Langdon and Justyna Petke. 2017. Software is Not Fragile. In *First Complex Systems Digital Campus World E-Conference 2015*, Paul Bourgine, Pierre Collet, and Pierre Parrend (Eds.). Springer International Publishing, Cham, 203–211.

[18] Joel Lehman and Kenneth O. Stanley. 2008. Exploiting Open-Endedness to Solve Problems Through the Search for Novelty. In *Proceedings of the Eleventh International Conference on the Synthesis and Simulation of Living Systems, ALIFE 2008, Winchester, United Kingdom, August 5-8, 2008*, Seth Bullock, Jason Noble, Richard A. Watson, and Mark A. Bedau (Eds.). MIT Press, 329–336. http://mitpress2.mit.edu/books/chapters/0262287196chap43.pdf

[19] Jian-Ping Li, Marton E. Balazs, Geoffrey T. Parks, and P. John Clarkson. 2002. A Species Conserving Genetic Algorithm for Multimodal Function Optimization. *Evol. Comput.* 10, 3 (sep 2002), 207–234. https://doi.org/10.1162/106365602760234081

[20] Jian-Ping Li and Alastair S. Wood. 2009. An adaptive species conservation genetic algorithm for multimodal optimization. *Internat. J. Numer. Methods Engrg.* 79, 13 (2009), 1633–1661. https://doi.org/10.1002/nme.2621 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2621

[21] H.R. Maier, S. Razavi, Z. Kapelan, L.S. Matott, J. Kasprzyk, and B.A. Tolson. 2019. Introductory overview: Optimization using evolutionary algorithms and other metaheuristics. *Environmental Modelling & Software* 114 (2019), 195–213. https://doi.org/10.1016/j.envsoft.2018.11.018

[22] Justyna Petke, Brad Alexander, Earl T. Barr, Alexander E. I. Brownlee, Markus Wagner, and David R. White. 2019. A Survey of Genetic Improvement Search Spaces. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Prague, Czech Republic) *(GECCO '19)*. Association for Computing Machinery, New York, NY, USA, 1715–1721. https://doi.org/10.1145/3319619.3326870

[23] Justyna Petke, Saemundur O Haraldsson, Mark Harman, William B Langdon, David R White, and John R Woodward. 2018. Genetic Improvement of Software: A Comprehensive Survey. *IEEE Trans. Evol. Comput.* 22, 3 (June 2018), 415–432.

[24] Barry Porter, Matthew Grieves, Roberto Rodrigues Filho, and David Leslie. 2016. RE^{X}: A Development Platform and Online Learning Approach for Runtime Emergent Software Systems. In *Symposium on Operating Systems Design and Implementation*. USENIX, 333–348.

[25] M.M. Raghuwanshi and O.G. Kakde. 2006. Genetic Algorithm With Species And Sexual Selection. In *2006 IEEE Conference on Cybernetics and Intelligent Systems*. 1–8. https://doi.org/10.1109/ICCIS.2006.252229

[26] Penny Faulkner Rainford and Barry Porter. 2022. Code and Data Synthesis for Genetic Improvement in Emergent Software Systems. *ACM Trans. Evol. Learn. Optim.* (May 2022).

[27] Penny Faulkner Rainford and Barry Porter. 2022. Lineage Selection in Mixed Populations for Genetic Improvement. In *ALIFE 2022: The 2022 Conference on Artificial Life*. MIT Press. https://doi.org/10.1162/isal_a_00494 arXiv:https://direct.mit.edu/isal/proceedings-pdf/isal2022/34/16/2035375/isal_a_00494.pdf

[28] Yago Saez, Cesar Estebanez, David Quintana, and Pedro Isasi. 2019. Evolutionary hash functions for specific domains. *Applied Soft Computing* 78 (2019), 58–69. https://doi.org/10.1016/j.asoc.2019.02.014

[29] Mustafa Safdari and Ramprasad Joshi. 2009. Evolving Universal Hash Functions Using Genetic Algorithms. In *2009 International Conference on Future Computer and Communication*. 84–87. https://doi.org/10.1109/ICFCC.2009.66

[30] Daan van Laar. 2021. *Fitness Landscape Analysis applied to functional Genetic Improvement.* Ph. D. Dissertation.

[31] John S Wilkins. 2009. *Defining species: a sourcebook from antiquity to today*. Vol. 203. Peter Lang.

[32] Ka-Chun Wong, Kwong-Sak Leung, and Man-Hon Wong. 2009. An Evolutionary Algorithm with Species-Specific Explosion for Multimodal Optimization. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation* (Montreal, Québec, Canada) *(GECCO '09)*. Association for Computing Machinery, New York, NY, USA, 923–930. https://doi.org/10.1145/1569901.1570027

[33] Sewall Wright. 1932. The roles of mutation, inbreeding, crossbreeding, and selection in evolution. In *Proc. 6th Int. Congress Genet., 1 (1932)*. na, 356–366.

[34] Feng Zou, Debao Chen, Hui Liu, Siyu Cao, Xuying Ji, and Yan Zhang. 2022. A survey of fitness landscape analysis for optimization. *Neurocomputing* 503 (2022), 129–139. https://doi.org/10.1016/j.neucom.2022.06.084