

Biological insights on grammar-structured mutations improve fitness and diversity

Stefano Tiso
s.tiso@rug.nl
University of Groningen
Groningen, The Netherlands

Pedro Carvalho
pfcarvalho@dei.uc.pt
University of Coimbra
Coimbra, Portugal

Nuno Lourenco
University of Coimbra
Coimbra, Portugal

Penousal Machado
University of Coimbra
Coimbra, Portugal

ABSTRACT

Grammar-Guided Genetic Programming (GGGP) employs a variety of concepts from evolutionary theory to autonomously design solutions for a given task. Recent insights from evolutionary biology can lead to further improvements in GGGP algorithms. In this paper, we propose a new mutation approach called Facilitated Mutation (FM) that is based on the theory of Facilitated Variation. We evaluate the performance of FM on the evolution of neural network optimizers for image classification, a relevant task in Evolutionary Computation, with important implications for the field of Machine Learning. We compare FM and FM combined with crossover (FMX) against a typical mutation approach to assess the benefits of the approach. We find that FMX provides statistical improvements in key metrics, creating a superior optimizer overall (+0.5% average test accuracy), improving the average quality of solutions (+53% average population fitness), and discovering more diverse high-quality behaviors (+523 high-quality solutions discovered on average). Additionally, FM and FMX reduce the number of fitness evaluations in an evolutionary run, reducing computational costs. FM's implementation cost is minimal and the approach is theoretically applicable to any algorithm where genes are associated with a grammar non-terminal, making this approach applicable in many existing GGGP systems.

CCS CONCEPTS

• **Software and its engineering** → **Genetic programming**; • **Computer systems organization** → *Neural networks.*

KEYWORDS

Grammar-guided Genetic Programming, Mutation, Optimizers

ACM Reference Format:
Stefano Tiso, Pedro Carvalho, Nuno Lourenco, and Penousal Machado. 2023. Biological insights on grammar-structured mutations improve fitness and diversity. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583131.3590472



This work is licensed under a Creative Commons Attribution International 4.0 License.

GECCO '23, July 15–19, 2023, Lisbon, Portugal
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0119-1/23/07.
https://doi.org/10.1145/3583131.3590472

1 INTRODUCTION

Genetic Programming (GP) is an Evolutionary Algorithm (EA) that evolves programs to solve a given task. A relevant branch of GP is Grammar-Guided Genetic Programming (GGGP) which uses a grammar to translate the representation (i.e., the genotype, a data structure representing the program) into an executable program (i.e., the phenotype or solution). In GGGP, practitioners can include domain knowledge by designing the grammar to bias the evolutionary search toward certain solutions. This advantage has led to successful applications of GGGP in many problem domains [12, 20, 24].

Despite these successes, GGGP can face difficulties exploring challenging solution spaces. Some GGGP approaches are prone to poor locality [36], meaning that small changes in the genotype lead to drastic changes in the phenotype. These problems are evident in tasks like the evolution of optimizers for neural network training. Carvalho et al. [8] use GGGP to evolve optimizers that compete with state-of-the-art human-designed solutions. While the result is favorable, the low average fitness of the population and the scarcity of reasonable solutions suggest that evolution is not exploring the solution space efficiently. Evolution appears to be randomly sampling the solution space (occasionally stumbling on promising genotypes) [47], rather than systematically improving upon incrementally fit solutions. Locality is often regarded as a matter of representation [36], but insights from evolutionary biology hint that variation operators may be part of the problem.

In nature, different parts of the genome display different amounts of susceptibility to mutation [34]. Core genes that have a vital function tend to mutate less [31], while genes with minor effects tend to mutate more often [49]. GGGP problems may share some of these dynamics with natural systems. Nevertheless, GGGP traditionally applies variation operators (mutation and crossover) uniformly across all genes without differentiating their functional role [32]. By not discriminating among the different types of genes, researchers have little control over the effect of variation on the phenotype and, thus, on the evolutionary trajectory of the system.

In this work, we propose a method to overcome the uncontrolled effects of mutations and allow a more faithful inheritance of traits by mimicking biological systems. We call our method Facilitated Mutation (FM), as it, in part, takes inspiration from the theory of Facilitated variation proposed by Kirschner and Gerhart in [15]. In FM, each grammar non-terminal uses a different mutation rate according to its predicted influence on the phenotype: low-impact

mutations are frequent, while high-impact mutations are rare. We expect FM to enable the evolutionary algorithm to explore the solution space in a more structured way, reducing reliance on random sampling. Note that FM is not a mutation operator, it is simply a novel way to apply existing mutation operators. Consequently, the implementation cost of FM is minimal and the efficacy is related to the mutation operator used.

We validate FM by evolving optimizers to train neural networks for image-recognition tasks. We compare our new mutational approach with the one used in past works [8]. Our contributions are as follows: 1) The best optimizers found by FM are statistically superior to the best optimizers found by competing approaches. 2) The average fitness of the population is statistically superior using FM. 3) FM statistically increases the diversity of discovered viable solutions. 4) FM reduces the computational cost of experiments.

The remainder of this manuscript will be structured as follows: In the Background, we provide the rationale from evolutionary biology guiding our design choices, as well as the current work in evolving optimizers and how it relates to larger GP and GGGP challenges. In the Methods, we describe the implementation of FM and how it applies to optimizer evolution. In the Experimental Setup, we describe our validation experiments and the metrics we use to compare FM against more traditional mutational approaches. In the Results, we compare results from two versions of FM against two versions of traditional mutation approaches on four metrics: performance of the best solution, average performance of the evolved solutions, diversity, and computational cost. Finally, in the Conclusions section, we summarize our findings and extend our reasoning to general consideration and future directions.

2 BACKGROUND

2.1 Insights from Biological Evolutionary Theory

Significant advancements in genomics, cell physiology, and developmental biology allowed evolutionary biology to gain an unprecedented understanding of the role of mutations and their effects on evolution. Of particular interest are insights on **heterogeneous mutation rates** and **heterogeneous mutation effects**.

Mutations that lead to phenotypic change do not happen homogeneously across the genome [13, 21]. There are conserved genes that code for fundamental *core processes*, that only mutate their function rarely, being more *robust* to mutation.[4, 40]. Many cellular and developmental processes are fundamentally unchanged since their emergence, such as the formation of micro-tubule structures [17, 48], cell adhesion processes [23, 25, 41], and anteroposterior axis formation [16]. Most mutations, instead, affect the phenotype by changing *regulatory elements* [39]. Changes in these elements do not alter the function of the core processes but regulate their activities and how they combine [43].

Not only do mutations not happen randomly across the genome, but they also have non-random effects on the phenotype depending on where they happen. If a mutation alters the functioning of a core process, it will fundamentally change the inner workings of the organism, most likely leading to a non-viable phenotype. Instead, mutations on regulatory sequences change the levels of expression of core processes and their interactions, producing functional phenotypes that are variations on the original theme.

Like biological systems, in evolutionary computation (EC) different genes play different roles in determining the final phenotype. However, unlike biological systems, EC often implements only a single mutation rate for all genes. By taking heterogeneous mutation rates and heterogeneous mutation effects into account EC systems might be able to display other interesting dynamics discussed by evolutionary biology: **developmental biases** and **facilitated variation**.

Empirical and theoretical research in evolutionary-developmental biology suggests that evolution has led to (and might have selected for) organisms whose core processes bias random mutations to produce a non-random subset of phenotypes. In the literature, this phenomenon is called developmental bias [42, 44]. Examples of developmental biases are: the number and distribution of digits, limbs, and segments in tetrapods [1], the structural and pigment coloration of insect wing [3], and flower morphology [45].

In line with the observations illustrated so far, the Theory of Facilitated Variation by Gerhart and Kirschner [15] postulates that organisms are structured to produce variation easily and that this variation is biased towards adaptive phenotypes. Organisms can do this because they contain an archive of evolved core processes that rarely change due to mutations and maintain functionality independently of context. Core processes can easily combine their activities through mutations on regulatory sequences and, since they maintain functionality across contexts, the outcome of these combinations is biased to be neutral or beneficial.

The predictions of this theory have profound implications for evolutionary dynamics. Facilitated variation can explain how evolution is dramatically faster than we expect since it selects core processes that favor the rapid discovery of functional solutions, instead of proceeding randomly [35]. By applying heterogeneous mutation rates and considering the heterogeneous effects of mutations, EC systems could behave in similar ways to what Facilitated variation predicts, possibly improving their efficiency and effectiveness. By slowing the mutation rate of core genes and increasing the rate of regulatory ones, evolution would explore only a few core processes at the time (low mutation rate), but in different combinations (high mutation rate). This would steer the search trajectory towards a reduced subset of all the possible solutions, giving time to select and improve the most successful ones before new ones are introduced. Thus, EC systems could incrementally build an archive of good solutions that can recombine together effectively, granting a better exploration of the solution space.

2.2 Mutation and Crossover in GGGP

In GGGP the genome of individuals encodes for a set of derived rules, defined by how its genome is translated by a grammar. These production rules can either be non-terminals (invoking more production rules) or terminals (constants or operations). Mutations in GGGP change the genome, potentially changing one or more of the chained production rules. A change in a production rule can lead to an important reshaping of the resulting behavior. Mutating a terminal into a non-terminal (which will invoke more production

rules) will expand the number of executed operations. Mutating a non-terminal into a terminal instead will cancel a series of operations, preventing them from ever being called. Thus, it is important to realize that, independently of which representation is used (e.g. Grammatical Evolution [GE], Structured Grammatical Evolution [SGE], etc.), GGGP mutations can cause different effects of different magnitudes depending on where they happen. GGGP often uses mutations in tandem with another variation operator: crossover. This is done to improve the exploration of the solution space. During crossover, a region of the genotype of one individual is swapped with a region from another one. The resulting individual is therefore a novel combination of the two parent genotypes. However, there is no guarantee that this new combination will resemble a combination of the two parents. Traditionally in GGGP, crossover methods are operated not by identifying functionally equivalent regions of the genome, but by simply exchanging non-terminals. Therefore, crossover in GGGP leads to highly diverse, but very unstructured variation.

As biology suggests, if core components are not protected, arbitrarily changing the genome structure (both through mutations and crossover) will likely lead to system-wide deleterious effects. Nonetheless, studies in GGGP normally use a uniform rate of mutation over the entire genome and rely heavily on crossover.

2.3 Evolution of optimizers showcases both potential and issues of GGGP

Optimizers are a relevant research topic in computer science as their application domains like Machine Learning, have deep practical implications. These algorithms can train Artificial Neural Networks to solve a target task using back-propagation [37]. Recent research shows that GP is a viable way to design competitive optimizers. GP systems such as PushGP[27–29] and AutoML-Zero[10] both have notable results in evolving optimizers. GGGP also achieves important results through the AutoLR framework [8], which is based on the sub-branch of GGGP known as Structured Grammatical Evolution (SGE) [30]. AutoLR is capable of evolving human-competitive optimizers from scratch without prior knowledge of the components used by human-made ones. However, despite the good results, some problems are evident in the way that AutoLR navigates the solution space. Very few solutions obtain a high fitness and most discovered solutions are non-functional. Moreover, high fitness solutions emerge mainly during early evolution and do not improve any further, suggesting that the evolutionary algorithm may be behaving similarly to random search[18, 19, 47]. Random search can be helpful to explore a solution space [2, 14, 38], but it does not fit the goals of Evolutionary Computation to incrementally produce diverse and competitive solutions. The shortcomings of AutoLR are likely caused by the fragility of the solutions to homogeneous mutation rates and indiscriminate use of crossover. Much like natural systems have core processes necessary for viable phenotypes, optimizers have core components (e.g., the gradient) that are likely to generate non-viable phenotypes when changed. But unlike natural systems, current optimizer evolution in GGGP does not consider heterogeneous mutation rates and mutation effects. These issues are not exclusive to AutoLR and GGGP. In fact, this is a well-known problem in GP [18, 19, 47], where algorithms often display low locality (changes in the genotype are disproportionally reflected on the phenotype) [30, 33].

Some researchers have already worked to mitigate this issue. Whigham [46] already in 1996 proposed the fine-tuning of heterogeneous mutation rates in GGGP. However, this is a small part of broader work from Whigham at the time, and only covers this idea briefly, not considering how biological evolution insights can inform design decisions in this subject. There is also work that explores forms of heterogeneous mutation outside of GP. Evolution Strategies (ES) are an entire type of EA that focuses heavily on adaptive control of parameters [22]. These systems adapt mutation rates and mutation steps for more successful mutation outcomes. However, these systems are commonly used for continuous search problems and do not address the same tasks as GGGP. Additionally, ES do not have grammars that can easily link different genes to different mutation rates in a structured, expertly informed way. As such, the possible benefits of a solution that mimics the patterns of natural mutations to address poor locality remains largely unexplored.

3 METHODS

3.1 Facilitated Mutation

The traditional GGGP approach applies mutationsuniformly across the genome not exploiting more fine-tuned mutations as natural systems do. In this work, we propose a new mutation approach for GGGP, Facilitated Mutation (FM), to replicate natural mutations.

Like in nature, mutations on different genes in GP produce different effects. FM uses heterogeneous mutation rates to support these differences. The goal of this endeavor is to lead to an efficient, incremental search of the solution space, bringing tangible benefits.

Specifically, FM leverages the structure of the existing grammar to create several tiers of mutations. Each grammar non-terminal uses a different mutation rate: higher for more impactful non-terminals, lower for less influential ones, as shown in Figure 1c. Separate mutation rates allow practitioners to use their expert knowledge to tune the rate of different mutation effects optimally.

It is important to notice that FM is not a mutation operator, the approach only regulates the application of the ones already in use.FM is consequently widely applicable since it does not require the implementation of new complex algorithms or operators. Instead, it only requires small adjustments to the way that mutation is applied in the system.

Moreover, biological insights suggest that swapping arbitrary regions of the genotype via crossover will more often than not disrupt the coordinated activity of genes, causing high-impact deleterious variation. Thus FM, when it employs it, sets crossover at low rates to allow time for new combinations of genes to fine-tune their concerted action.

3.2 Facilitated Mutation in Optimizer Evolution

In this work, we apply the ideas of FM to AutoLR [8], a framework for the evolution of neural network optimizers based on a GGGP engine called SGE [30]. The evolution of optimizers is an excellent task to evaluate FM. Optimizers have clearly distinguishable core components (e.g., gradient) crucial to their functioning, and regulatory components (e.g., tuning parameter values) dedicate to

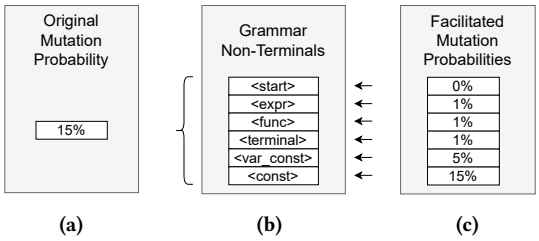


Figure 1: Traditionally, mutation operators use a single mutation rate (1a) for all grammar non-terminals (1b). In FM, there is a separate mutation rate for each non-terminal (1c)

their fine-tuning. These distinct components would benefit from different mutation rates.

However, the original AutoLR grammar combines some crucial and tuning components in the same non-terminals. Specifically, the gradient shares a non-terminal with other, less important ones. The gradient is the most important component of an optimizer, any change to the gradient in an active gene will likely result in a complete overhaul of the solutions' behavior. To improve the efficacy of FM, we create a new grammar that separates components with different roles in different non-terminals. The new grammar still maintains the original grammar's functionality, as it uses the same components. In Figure 2, we show the two grammars, side-by-side. The complete grammars can be found at [5, 7]We refer the reader to the AutoLR paper [8] for details on the design decisions, operations, and variables of the original grammar.

Leveraging these new non-terminals, we set specific mutation rates based on expert knowledge. We assign a 0.15 mutation rate to constants, allowing easy tuning of parameters. We assign a 0.05 mutation rate for changes between constants and other variables. We assign a 0.01 mutation rate to all other non-terminals. This 0.01 mutation rate affects changes that influence the behavior of the optimizer the most i.e.: changes from the gradient to anything else, as well as changes in the operations between variables. Figure 1c presents the complete set of parameters. Finally, we allow FM to make use of crossover (FM with crossover: FMX), but at a rate that is significantly smaller than previous setups (FMX crossover rate = 0.01 << original crossover rate = 0.9). We chose this crossover rate to be just high enough to avoid stagnation on local minima but not high enough to disrupt solutions that emerge from mutation.

4 EXPERIMENTAL SETUP

We validate Facilitated Mutation by using AutoLR to evolve optimizers that train a Convolutional Neural Network for image classification. Optimizers are assigned a fitness value based on the classification accuracy achieved by the network they trainand undergo tournament selection of size two.Winners of the tournament reproduce with a probability of undergoing mutation and crossover.This procedure is repeated until the population for the next generation is filled up. The population size is 100 individuals, and evolution runs for 200 generations. The best solution of each generation is preserved via elitism . We run 30 independent simulations for four different mutation approaches:

<start> ::=<expr>
<expr> ::=<func> | <term>
<func> ::=<expr>+<expr> |
<expr>*<expr> |
...
<term> ::=<const> |
alpha | grad |
...
<const> ::= 0.0 | $5E^{-5}$ |
...

(a) Original Grammar.

<start> ::=<expr>
<expr> ::=<func> | <term>
<func> ::=<expr>+<expr> |
<expr>*<expr> |
...
<term> ::=<var_const> |
grad | ...
<var_const> ::=<const> |
alpha | ...
<const> ::= 0.0 | $5E^{-5}$ |
...

(b) Grammar for Facilitated Mutation.

Figure 2: The original grammar used in AutoLR (2a) and the adapted version for Facilitated Mutation (2b). Non-terminals that have been added or changed are underlined.

Table 1: Evolutionary parameters for the four different setups

Parameters	Value			
	FMX	FM	OM	OMX
Crossover Rate	0.01	0.0	0.0	0.9
Mutation Rate	Heterogeneous		Homogeneous (0.15)	
Population Size	100			
Generations	200			
Elitism	1%			
Tournament Size	2			
Max Depth	17			

(1) **Facilitated Mutation with rare crossover (FMX)**: our novel solution implementing both changes to mutation rates and to crossover (*low rate*).
(2) **Facilitated Mutation (FM)**: our novel solution in isolation to investigate its effectiveness in the absence of crossover.
(3) **Original mutation without crossover (OM)**: the original solution without crossover, for direct comparisons with FM.
(4) **Original mutation with crossover (OMX)**: the original setup used in past works [8].

Table 1 describes the evolution parameters for each approach. We use four metrics to analyze and compare the results from the different mutational approaches (FMX, FM, OM, OMX):

- **Best Fitness**: The *fitness of the best evolved solution* in a simulation.
- **Population Fitness**: The *average fitness of the population.* We also consider the *distribution of fitness in the population.*
- **Population Diversity**: The *number of different viable and distinct solutions in a population.* We distinguish solutions by the equation encoding the optimizer. We set a threshold of accuracyto distinguish non-viable (< 0.5) and viable (> 0.5) solutions.

- **Computational Cost**: *Number of training events over the evolutionary run.* Most computational costs in AutoLR come from the training necessary for the fitness evaluation.AutoLR avoids re-evaluating solutions it has already encountered (see 4.2); thus, algorithms that bias variation can reduce computational costs.

4.1 Task, training and fitness evaluation

We evolve the optimizers to train a simple convolutional neural network [6] for image classification on Fashion-MNIST [50]. Fashion-MNIST is a data set with 60000 28x28 gray-scale training images. Each image depicts an article of clothing belonging to one of ten categories (e.g., T-shirt/top, trousers, pullover). The optimizer trains the network to label these images correctly. Training has two stop conditions. First, the procedure terminates when a maximum number of epochs is reached. Second, an early stop mechanism terminates the procedure if there is no progress for a set number of consecutive epochs. To ensure a fair fitness assessment, we split Fashion-MNIST into three data sets: Training Data - During training, the network assigns labels to this data and the optimizer changes the network's weights based on the errors of the assigned label. The optimizer directly interacts with this data. Validation Data - During training, validation metrics are calculated on this data to monitor progress. The early stop mechanism monitors these validation metrics to terminate training when no progress is made. The optimizer does not interact directly with this data. Fitness Data - After training concludes, accuracy is calculated using this data and used as the optimizer's fitness. The optimizer does not directly interact with this data, ensuring an unbiased fitness assessment. A table summarizing the parameter values used in our Fashion-MNIST experiments is presented in the supplementary material (Supplementary Material Table 1).

4.2 Solution archive and viable solution pre-selection

Once fitness is calculated, the optimizer and its fitness are recorded in an archive. Before evaluating any optimizer, the system will check the archive for an existing record to avoid re-evaluation, since multiple genomes can encode for the same optimizer. Additionally, the system can detect dysfunctional optimizers before evaluation: an optimizer that does not use the gradient cannot train a neural network, so AutoLR assigns them a poor fitness value (0.1) and skips evaluationThese two procedures together save considerable computational resources.

4.3 Post-hoc Analysis

While the fitness function provides a quick assessment of the quality of an optimizer, it is ultimately an abridged procedure that focuses on efficiency. We use limited training size, few training epochs, and an early stop mechanism to speed up computation during the evolutionary simulations. However, these expedients limit the thoroughness of the training and thus hinder the assessment of the full capabilities of an optimizer. We perform a post-hoc analysis on the best evolved solution for each approach to assess their quality more accurately. The procedure for post-hoc analysis is similar to the fitness evaluation, with a few key differences. Training uses 56500 examples instead of 6500 and continues for 1000 epochs instead of 100. Moreover, the early stop mechanism is removed. After training, the weights that achieved the best validation-accuracy are restored for a final test-accuracy assessment. The system calculates the test-accuracy using 10000 images from the Fashion-MNIST test data. This test data is a separate data set, never used during evolution, thus ensuring an accurate assessment of the optimizers' ability to generalize to new data. We repeat the entire procedure 15 times, training the network from scratch and recalculating the test accuracy in each repetition. After all these steps, the average test-accuracy of an optimizer is the best measure of its actual quality.

Table 2: Results summary

Approach	Metrics			
	Best Fitness	Population Fitness	Population Diversity	Computational Cost
FMX	**0.933[L]**	**0.625[L]**	**599[M]**	2915
FM	0.931	**0.608[L]**	323	**2137[M]**
OM	0.927	0.281	122	2978
OMX	0.928	0.093	73	10933

Values for the four metrics across approaches. Best results are in bold with Cohen's d [11] effect size as "S": small; "M":medium; and "L":large

5 RESULTS

In this section, we report and analyze the results obtained from the 30 runs of each of the four mutation approaches (FMX, FM, OMX, OM). For a summary of the results, see Table 2, best results for each metric are highlighted in bold. We conduct a statistical comparison between the different mutational approaches using the t-test (significance level set to $\alpha < 0.05$) and calculate the effect size and p-value [11]. A full overview can be found in the supplementary material (Supplementary Material Tables 2 and 3). In this section's figures, if there are statistically significant differences displayed, we use asterisks to show the significance level and omit the full p-values ($0.05 > p > 0.01$ = *, $0.01 \geq p \geq 0.001$ = **, $0.01 \geq p \geq 0.001$ = ***, $p < 0.0001$ = ****). The four different metrics: **Best fitness**, **Population fitness**, **Population diversity** and **Computational cost** are discussed separately in different subsections.

5.1 Best Fitness

We perform post-hoc analysis on the single best optimizer out of the 30 different runs from each approach and compare their final quality. The result of the test and validation accuracy assessments, as well as the p-value results from pairwise optimizer comparisons, are shown in Figure 3. Looking at the results, one can see that the approaches that rely on FM produce the best results overall.

After post-hoc analysis, FMX produced the most effective optimizer, classifying 93.3% of test images correctly on average. FMX's optimizer is statistically superior to all other approaches in test and validation-accuracy. The FM and OMX optimizers are comparable in test-accuracy, despite FM being superior in validation-accuracy. OM evolved the weakest optimizer. These results confirm that FM aids evolution in producing superior optimizers. Both OM and FM

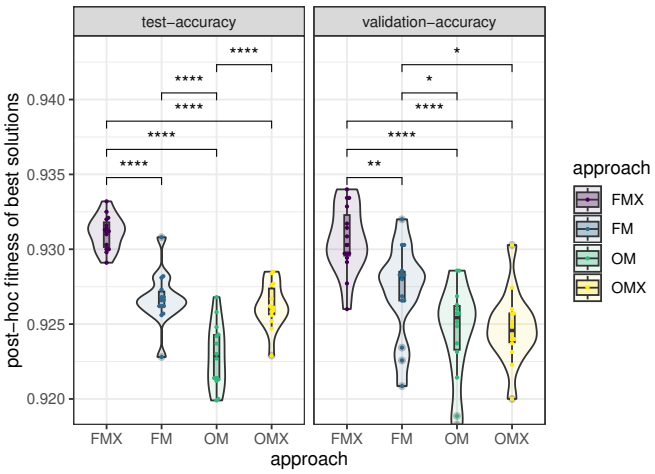


Figure 3: Test and validation accuracies of the best optimizer for each setup in post-hoc analysis.

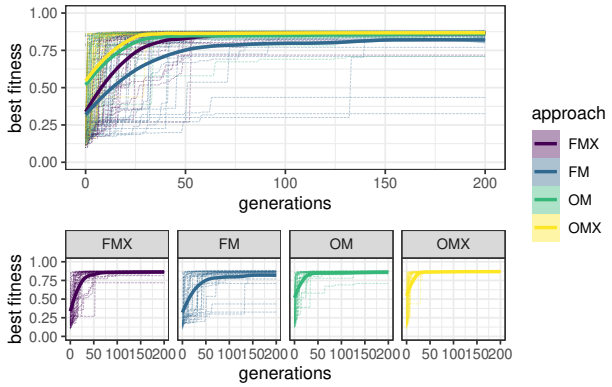


Figure 4: Fitness of the best individual of the 30 populations over time for the four different approaches. Bold lines show mean values in each approach.

improve when combined with crossover, suggesting that this operator contributes to discovering the best solutions despite its apparent destructive effects.

We also analyzed the consistency of the approaches, comparing the best solutions found during evolution by all 30 runs of each approach (shown in Figure 4). FMX, OM, and OMX have comparable consistency in discovering high-fitness solutions. OM and OMX methods plateau rapidly as their mutational approaches explore the solution space very fast but impair incremental change, meanwhile FMX reaches a comparable level more gradually by steadily improving upon discovered solutions. FM is the most inconsistent, with a substantial number of runs below 0.5 fitness for the best solution. This is due to the absence of crossover combined with the prevalence of neutral mutations, and the use of a solution archive. Without crossing over, FM mutation can only navigate the solution space via frequent regulatory mutations that do not affect the solution i.e.: neutral mutations. This evolutionary pattern is called neutral search, and it allows solutions to move across the solution space while being shielded from selection. Neutral search is computationally efficient in AutoLR because the system only spends time evaluating new solutions and not new genotypes. However, since runs for all mutational approaches terminate after a fixed number of generations, FM will explore fewer solutions than the others. Therefore, FM is penalized by AutoLR as it has fewer chances of discovering promising optimizers and is more prone to get stuck into sub-optimal regions of the solution space. On the other hand, FMX can discover higher-performance solutions thanks to the low rate of crossing over, which nudges the population into unexplored regions of the solution space, where new solutions can be evaluated. OMX and OM need not rely on the crossover as their highly disruptive mutation approach is sufficient to move them to new regions of the solution space.

5.2 Population Fitness

FM and FMX substantially improve the average solution fitness in the population. At any point in time, FM and FMX populations have higher average fitness than OM/OMX (Figure 5). FMX and FM have an average population fitness of ≈ 0.6, OM and OMX have average fitnesses of ≈ 0.3 and ≈ 0.1, respectively, as Figure 6 shows. We find that this difference is statistically significant at the end of evolution.

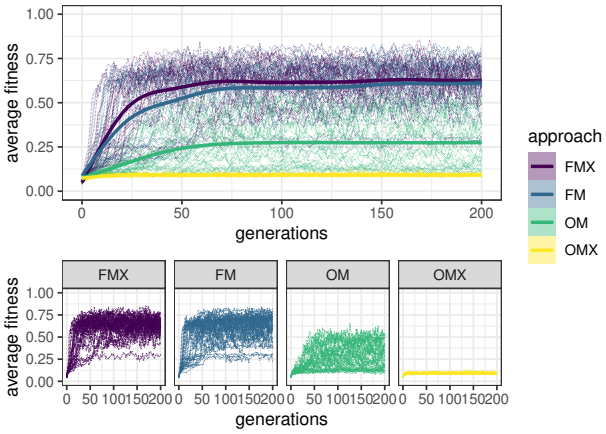


Figure 5: Average fitness of the 30 populations over time for the four different approaches.

In OMX and OM, average solutions in a population have a fitness one order of magnitude smaller than the best one (OMX and OM average solution fitness ≈ 0.1, OMX and OM best solution fitness ≈ 0.9). This is in accordance with results from previous studies[8]. In contrast to FMX, OMX proceeds by randomly sampling good solutions from a vast solution space thanks to its uniform mutation rate and high crossover rate (0.9) but does not traverse the search space efficiently, inhibiting fit solutions from producing incrementally fitter offspring. These results demonstrate the delicate role of crossover rate in balancing exploration and exploitation and the importance of tuning this parameter correctly. A high crossover rate has a highly negative impact on population fitness. Population

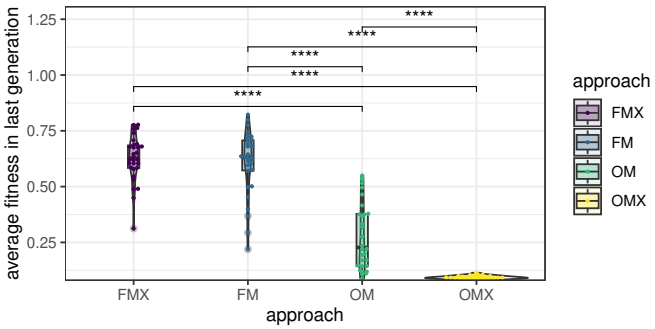


Figure 6: Average fitness in the last generation of each run for all approaches.

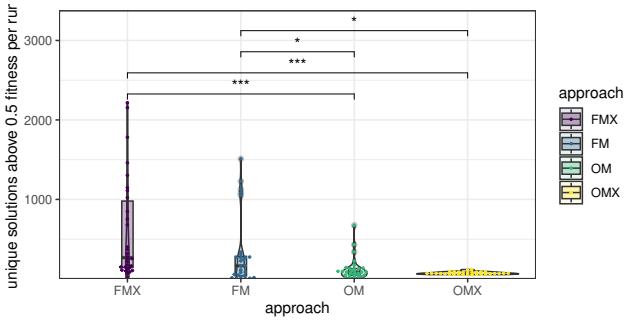


Figure 7: Unique behaviors above 0.5 fitness discovered per run. Higher values mean more diversity.

fitness in OMX (crossover rate = 0.9) is significantly lower than its crossover-free counterpart OM (crossover rate = 0). This difference contrasts with the positive effect that small crossover rates can have, as observed in Section 5.1. FMX and FM population fitnesses do not show the same drop. The difference in the response to crossover may caused by the different crossover rates (FMX crossover rate = 0.01 << OMX crossover rate = 0.9). Further study into how each approach reacts to changes in crossover rate is necessary to confirm this idea since such experiments are outside the scope of this work.

Improved population fitness supports the hypothesis that FM and FMX approaches can improve the navigation of the solution space. However, analyzing the population's fitness alone is not sufficient. Many identical copies with high fitness could lead to the same result. It is essential to analyze the diversity of solutions discovered by each approach to asses if FM and FMX approaches can explore the solution space better.

5.3 Population Diversity

We measure population diversity as the *number of viable unique behaviors in the population*. The metric we use is a modified version of the *variety* measure proposed by Koza [26]. This measure is defined as "the percentage of individuals for which no exact duplicate exists elsewhere in the population" we adapt this to work on *number of unique behaviors* rather than percentages but retain the focus on "individuals for which no exact duplicate exists elsewhere in the population" which we abbreviate to "unique behaviors in the population". We could use an edit-distance based measure instead, but we find that a simple, behavior focused measure is the best match for this study.

Optimizers are fragile and non-viable solutions are frequent, thus including non-viable solutions in our analysis would confound information about the diversity of optimizers that actually contribute to evolution. We find "the number of *viable* unique behaviors" by considering solutions with a fitness above the threshold of 0.5, that are able to train the network in some capacity. We statistically compare the total number of unique viable behaviors discovered per run (Figure 7, for the number of unique behaviors regardless of fitness see Supplementary Material Figure 1).

FMX produces a significantly higher diversity of viable solutions than all other approaches (on average 600 viable unique solutions per run). FM produces on average 300 and OM 120, although this difference is not statistically significant, while OMX is statistically worst than all other approaches with just 70 unique viable solutions per run on average (Figure 7).

The lack of diversity in OMX clues us into the failures of this approach and confirms our hypothesis that FMs and FMX can explore the solution space more efficiently. Despite the vast amount of new behaviors that crossover introduces into the population, OMX cannot convert this into functioning behaviors that aid evolution. The combined insights about diversity and population fitness suggest that FMX and FM can keep a diverse, viable population longer than OM and OMX. In sum, these experiments suggest that FM and FMX succeed in the stated goal of improving the search for the solution space, possibly allowing them to avoid stagnation.

However, the results raise questions about performance. We previously hypothesized that FM might be computationally cheaper by evaluating fewer solutions thanks to an increase in neutral search. This analysis of diversity suggests that FM evaluates more viable solutions. One possible explanation for this result is that FM is saving resources on evaluations that are not reflected in our measure of diversity. Specifically, FM may reduce the number of evaluations below the threshold used to analyze diversity. To clarify this aspect, we analyze the computational cost of the experiments directly.

5.4 Computational Costs

Most of the computational costs of AutoLR come from the neural network training necessary for the fitness evaluation. In Figure 8, we statistically compare the number of training and evaluations (the main computational cost) executed in all approaches. AutoLR uses a solution archive and pre-selects only for functional optimizers to avoid redundant and irrelevant evaluations. Due to these mechanisms, the number of optimizers evaluated in a run is the number of unique behaviors that include at least one gradient component. All approaches we have implemented for the first time in this work (FMX, FM, OM) are statistically cheaper in terms of computational cost than previously implemented ones (OMX), see Figure 8). There is no statistically significant difference between FMX, FM, and OM (although FM sits just outside the established significance threshold when compared to OM, with a p-value = 0.057, see Supplementary Material Table 3). Moreover, the results

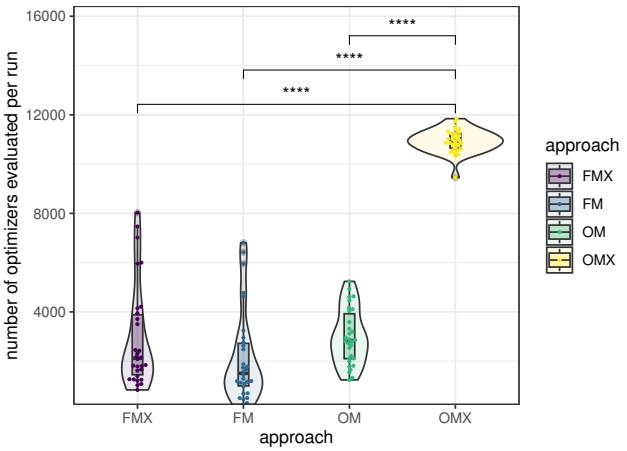


Figure 8: Total number of evaluated optimizers, for all approaches. Lower values mean less computational costs.

seem to indicate that FM is empirically cheaper than all the other approaches, but this difference is not statistically significant. FM thus seems to manage to be cheaper while at the same time having a high diversity of good optimizers (see Section 5.3). This suggests that FM performs neutral search (saving computational resources) in high-fitness regions of the problem space.

6 CONCLUSIONS

This paper proposes Facilitated Mutation, a mutational approach for GGGP inspired by biological evolution. Facilitated mutation splits the single mutation rate commonly used in these approaches into different values for each grammar non-terminal. We compare Facilitated Mutation approaches with (FMX) and without crossover (FM) against more traditional mutational approaches with (OMX) and without (OM) crossover. We validate these approaches on optimizer evolution, an evolutionary machine learning task with a challenging solution space, and real-world applications, and compare them based on the best optimizer discovered, average population fitness, population diversity , and computational cost .

FMX is comparable or better to original mutation approaches in all aspects. When compared to the best OM-based alternative: FMX evolved a better optimizer (+0.5% network test accuracy than OMX), and on average: produced better and discovered more solutions (+34% population fitness, +477 viable solutions) than OM These benefits come with no significant downsides, the approach has similar computational costs to OM and is significantly cheaper than OMX (saving $\approx$ 118000 fitness evaluations).

Comparing the two mutation approaches without crossover (FM vs. OM), we find that FM discovered a better optimizer (+0.4% network test accuracy), improved population fitness by +32%, discovered 200 additional viable individuals and decreased computational costs by 850 fitness evaluations on average.

While we only study its benefits in the context of an SGE-based framework, FM is applicable to any GGGP where each gene is directly associated with a grammar non-terminal. FM is simple in implementation and applicable to any GP problem, but the effectiveness depends on the type of problem and the choice of proper grammar and mutation rates.

6.1 Future Work

FM and FMX achieve several remarkable results in this work, but these mutational approaches can still be pushed further.

In this work, the rates of mutation and crossover have not been further fine-tuned after their initial choice. There are many ways to further optimize mutation rates for FM; most notably, we think a promising direction is to let the mutation rates themselves evolve. Since the submission of this work, we have created a new version of FM with adaptive mutation rates and outlined a grammar design procedure to amplify the benefits of this type of mutation [9].

Our analysis of crossover in this work can also be expanded. We focused on assessing the benefits of our solution through an informed decision of crossover rate (FMX crossover rate = 0.01) and drawing comparisons with another work with a comparable setup (OMX crossover rate = 0.9). However, we do not investigate how the different mutation approaches respond to different crossover rates. Crossover significantly improved the best fitness of both approaches so it is essential to study how these methods respond to varying crossover rates.

The analysis of the different mutational approaches can also be expanded to understand all their implications better. While we found viable measures for all aspects of the evolution we studied, technical limitations stopped us from looking at other metrics, such as solution complexity, the phenotypic distance between solutions, and the total run time of experiments.

Other aspects of evolution could also be studied, such as detecting optimizers that evolution discovers repeatedly and how these vary depending on the approach. The improved diversity and population fitness of FM approaches also warrant further research into whether the approach can also improve the system's evolvability and adaptation to different networks and tasks through transfer learning. In particular we find important to evaluate the performance of our approach in more challenging image recognition data sets (e.g., CIFAR-10). From our first analyses it appears that the performance for the Fashion-MNIST data set is saturated (see 4). FM methods may show an even larger margin of improvement in more complex data sets, where simpler mutational regimes (OM) could not navigate the higher complexity of the search space.

FM may improve performance even further for structured problems (e.g., robot control, path finding) where there is a limited set of possible actions that can be fully defined by the grammar. This contrasts with optimizer evolution, where the set of variables and operations used for evolution is arbitrarily restricted, infinitely varied and endlessly debatable.

Finally, previous works compared the evolved optimizers with human-made solutions. We did not perform this type of analysis since the focus of this work was on improving evolution, but such insights would be interesting in an extended version of this paper.

ACKNOWLEDGMENTS

We thank the members of the modelling adaptive response mechanisms (MARM) group at the University of Groningen for stimulating

discussion and comments. S.T., acknowledge funding from the European Research Council (ERC Advanced Grant No. 789240). This work was also funded by FEDER funds through the Operational Programme Competitiveness Factors - COMPETE and national funds by FCT - Foundation for Science and Technology (POCI-01-0145-FEDER-029297, CISUC - UID/CEC/00326/2020) and within the scope of the project A4A: Audiology for All (CENTRO-01-0247-FEDER-047083) financed by the Operational Program for Competitiveness and Internationalisation of PORTUGAL 2020 through the European Regional Development Fund. The second author is funded by FCT, Portugal, under the grant UI/BD/151053/2021.

REFERENCES

[1] P. Alberch and E. A. Gale. 1985. A developmental analysis of an evolutionary trend: digital reduction in amphibians. *Evolution* 39, 1 (1 1985), 8–23. https://doi.org/10.1111/J.1558-5646.1985.TB04076.X

[2] James Bergstra and Yoshua Bengio. 2012. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* 13, null (feb 2012), 281–305. https://dl.acm.org/doi/10.5555/2188385.2188395

[3] P. M. Brakefield and J. C. Roskam. 2006. Exploring Evolutionary Constraints Is a Task for an Integrative Evolutionary Biology. *The American Naturalist* 168, S6 (2006), S4–S13. https://doi.org/10.1086/509049 arXiv:https://doi.org/10.1086/509049 PMID: 17109328.

[4] Mónica Campillos, Christian Von Mering, Lars Juhl Jensen, and Peer Bork. 2006. Identification and analysis of evolutionarily cohesive functional modules in protein networks. *Genome Research* 16, 3 (3 2006), 374–382. https://doi.org/10.1101/GR.4336406

[5] Pedro Carvalho. 2023. AutoLR Facilitated Mutation Grammar. https://github.com/soren5/autolr/blob/master/grammars/adaptive_autolr_grammar_mutate_level.txt

[6] Pedro Carvalho. 2023. AutoLR MNIST Model. https://github.com/soren5/autolr/blob/master/models/json/mnist_model.json

[7] Pedro Carvalho. 2023. AutoLR Original Grammar. https://github.com/soren5/autolr/blob/master/grammars/adaptive_autolr_grammar_original.txt

[8] Pedro Carvalho, Nuno Lourenço, and Penousal Machado. 2022. Evolving Adaptive Neural Network Optimizers for Image Classification. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 13223 LNCS (2022), 3–18. https://doi.org/10.1007/978-3-031-02056-8{_}1

[9] Pedro Carvalho, Jessica Mégane, Nuno Lourenço, and Penousal Machado. 2023. Context Matters: Adaptive Mutation for Grammars. In *Genetic Programming: 26th European Conference, EuroGP 2023, Held as Part of EvoStar 2023, Brno, Czech Republic, April 12–14, 2023, Proceedings*. Springer, 117–132.

[10] Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Yao Liu, Kaiyuan Wang, Cho-Jui Hsieh, Yifeng Lu, and Quoc V Le. 2022. Evolved Optimizer for Vision. In *First Conference on Automated Machine Learning (Late-Breaking Workshop)*. https://openreview.net/forum?id=jK_eS5BxOuu

[11] Jacob Cohen. 2013. *Statistical power analysis for the behavioral sciences*. Routledge.

[12] Alfonso Ortega de la Puente, Rafael Sánchez Alfonso, and Manuel Alfonseca Moreno. 2002. Automatic Composition of Music by Means of Grammatical Evolution. In *Proceedings of the 2002 Conference on APL: Array Processing Languages: Lore, Problems, and Applications* (Madrid, Spain) *(APL '02)*. Association for Computing Machinery, New York, NY, USA, 148–155. https://doi.org/10.1145/602231.602249

[13] Lars Feuk, Andrew R. Carson, and Stephen W. Scherer. 2006. Structural variation in the human genome. *Nature Reviews Genetics 2006 7:2* 7, 2 (2 2006), 85–97. https://doi.org/10.1038/nrg1767

[14] Alexander Gajewski, Kenneth O. Stanley, Jeff Clune, and Joel Lehman. 2019. Evolvability ES: Scalable and direct optimization of evolvability. In *GECCO 2019 - Proceedings of the 2019 Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, Inc, 107–115. https://doi.org/10.1145/3321707.3321876

[15] John Gerhart and Marc Kirschner. 2007. The theory of facilitated variation. *Proceedings of the National Academy of Sciences of the United States of America* 104, SUPPL. 1 (5 2007), 8582–8589. https://doi.org/10.1073/pnas.0701035104

[16] John Gerhart, Christopher Lowe, and Marc Kirschner. 2005. Hemichordates and the origin of chordates. *Current Opinion in Genetics & Development* 15, 4 (2005), 461–467. https://doi.org/10.1016/j.gde.2005.06.004

[17] John C Gerhart and Marc W Kirschner. 1997. *Cells, Embryos and Evolution*. Blackwell Science 1997.

[18] Jens Gottlieb and Christoph Eckert. 2000. A Comparison of Two Representations for the fixed charge transportation problem. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 1917 (2000), 345–354. https://doi.org/10.1007/3-540-45356-3_34

[19] Jens Gottlieb and Günther R. Raidl. 2000. Locality in decoder-based EAs for the multidimensional knapsack problem. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 1829 (2000), 38–52. https://doi.org/10.1007/10721187_3

[20] Frederic Gruau. 1994. Automatic definition of modular neural networks. *Adaptive behavior* 3, 2 (1994), 151–183. https://doi.org/10.1177/105971239400300202

[21] Jonathan D. Gruber, Kara Vogel, Gizem Kalay, and Patricia J. Wittkopp. 2012. Contrasting properties of gene-specific regulatory, coding, and copy number mutations in saccharomyces cerevisiae: Frequency, effects, and dominance. *PLoS Genetics* 8, 2 (feb 2012). https://doi.org/10.1371/journal.pgen.1002497

[22] Nikolaus Hansen, Dirk V Arnold, and Anne Auger. 2015. Evolution strategies. *Springer handbook of computational intelligence* (2015), 871–898.

[23] Andrea Hartsock and W. James Nelson. 2008. Adherens and tight junctions: Structure, function and connections to the actin cytoskeleton. *Biochimica et Biophysica Acta (BBA) - Biomembranes* 1778, 3 (2008), 660–669. https://doi.org/10.1016/j.bbamem.2007.07.012

[24] Martin Hemberg, Una-May O'Reilly, Achim Menges, Katrin Jonas, Michel da Costa Gonçalves, and Steven R Fuchs. 2008. Genr8: Architects' experience with an emergent design tool. In *The Art of Artificial Evolution*. Springer, 167–188. https://doi.org/10.1007/978-3-540-72877-1_8

[25] Harald Hutter, Bruce E Vogel, John D Plenefisch, Carolyn R Norris, Rui B Proenca, John Spieth, Chaobo Guo, Surjeet Mastwal, Xiaoping Zhu, Jochen Scheel, et al. 2000. Conservation and Novelty in the Evolution of Cell Adhesion and Extracellular Matrix Genes. *Science* 287, 5455 (2000), 989–994. https://doi.org/10.1126/science.287.5455.989

[26] John R. Koza. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA. https://doi.org/10.1007/BF00175355

[27] Michael A Lones. 2019. Instruction-level design of local optimisers using push GP. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 1487–1494. https://doi.org/10.1145/3319619.3326806

[28] Michael A. Lones. 2020. Optimising Optimisers with Push GP. In *Lecture Notes in Computer Science*. Springer International Publishing, 101–117. https://doi.org/10.1007/978-3-030-44094-7_7

[29] Michael A Lones. 2021. Evolving continuous optimisers from scratch. *Genetic Programming and Evolvable Machines* 22, 4 (2021), 395–428. https://doi.org/10.1007/s10710-021-09414-8

[30] Nuno Lourenço, Francisco B. Pereira, and Ernesto Costa. 2016. Unveiling the properties of structured grammatical evolution. *Genetic Programming and Evolvable Machines* 17, 3 (9 2016), 251–289. https://doi.org/10.1007/S10710-015-9262-4/FIGURES/21

[31] Iñigo Martincorena and Nicholas M. Luscombe. 2013. Non-random mutation: the evolution of targeted hypermutation and hypomutation. *Bioessays* 35, 2 (2 2013), 123–130. https://doi.org/10.1002/bies.201200150

[32] Robert I McKay, Nguyen Xuan Hoai, Peter Alexander Whigham, Yin Shan, and Michael O'neill. 2010. Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines* 11, 3 (2010), 365–396. https://doi.org/10.1007/s10710-010-9109-y

[33] Eric Medvet. 2017. A comparative analysis of dynamic locality and redundancy in grammatical evolution. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10196 LNCS (2017), 326–342. https://doi.org/10.1007/978-3-319-55696-3{_}21/COVER

[34] J. Grey Monroe, Thanvi Srikant, Pablo Carbonell-Bejerano, Claude Becker, Mariele Lensink, Moises Exposito-Alonso, Marie Klein, Julia Hildebrandt, Manuela Neumann, Daniel Kliebenstein, Mao-Lun Weng, Eric Imbert, Jon Ågren, Matthew T. Rutter, Charles B. Fenster, and Detlef Weigel. 2022. Mutation bias reflects natural selection in Arabidopsis thaliana. *Nature 2022* (1 2022), 1–5. https://doi.org/10.1038/s41586-021-04269-6

[35] Merav Parter, Nadav Kashtan, and Uri Alon. 2008. Facilitated Variation: How Evolution Learns from Past Environments To Generalize to New Environments. *PLoS Computational Biology* 4, 11 (2008), 1000206. https://doi.org/10.1371/journal.pcbi.1000206

[36] Franz Rothlauf and Marie Oetzel. 2006. On the locality of grammatical evolution. In *Genetic Programming: 9th European Conference, EuroGP 2006, Budapest, Hungary, April 10-12, 2006. Proceedings 9*. Springer, 320–330.

[37] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. *Learning internal representations by error propagation*. Technical Report. California Univ San Diego La Jolla Inst for Cognitive Science. 318–362 pages. https://ieeexplore.ieee.org/document/6302929

[38] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *Open Ai* (2017), 1–13. arXiv:1703.03864 http://arxiv.org/abs/1703.03864

[39] David L. Stern and Virginie Orgogozo. 2009. Is Genetic Evolution Predictable? *Science* 323, 5915 (2009), 746–751. https://doi.org/10.1126/science.1158997 arXiv:https://www.science.org/doi/pdf/10.1126/science.1158997

[40] Amos Tanay, Aviv Regev, and Ron Shamir. 2005. Conservation and evolvability in regulatory networks: The evolution of ribosomal regulation in yeast. *Proceedings*

of the National Academy of Sciences of the United States of America 102, 20 (5 2005), 7203–7208. https://doi.org/10.1073/pnas.0502521102

[41] Ulrich Tepass, Guy Tanentzapf, Robert Ward, and Richard Fehon. 2001. Epithelial Cell Polarity and Cell Junctions in Drosophila. *Annual Review of Genetics* 35, 1 (2001), 747–784. https://doi.org/10.1146/annurev.genet.35.102401.091415 arXiv:https://doi.org/10.1146/annurev.genet.35.102401.091415 PMID: 11700298.

[42] Tobias Uller, Armin P. Moczek, Richard A. Watson, Paul M. Brakefield, and Kevin N. Laland. 2018. Developmental bias and evolution: A regulatory network perspective. *Genetics* 209, 4 (2018), 949–966. https://doi.org/10.1534/genetics.118.300995

[43] Günter P. Wagner, Mihaela Pavlicev, and James M. Cheverud. 2007. The road to modularity. *Nature Reviews Genetics* 8, 12 (2007), 921–931. https://doi.org/10.1038/nrg2267

[44] Richard A. Watson and Eörs Szathmáry. 2016. How Can Evolution Learn? *Trends in Ecology and Evolution* 31, 2 (2016), 147–157. https://doi.org/10.1016/j.tree.2015.11.009

[45] Carolyn A. Wessinger and Lena C. Hileman. 2016. Accessibility, constraint, and repetition in adaptive floral evolution. *Developmental Biology* 419, 1 (11 2016), 175–183. https://doi.org/10.1016/J.YDBIO.2016.05.003

[46] Peter Alexander Whigham. 1996. Grammatical Bias for Evolutionary Learning. (1996). http://hdl.handle.net/1959.4/60693

[47] Peter A. Whigham, James MacLaurin, Grant Dick, and Caitlin A. Owen. 2015. Examining the "best of both worlds" of grammatical evolution. In *GECCO 2015 - Proceedings of the 2015 Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, Inc, 1111–1118. https://doi.org/10.1145/2739480.2754784

[48] Rasmus G. Winther. 2001. Varieties of modules: Kinds, levels, origins, and behaviors. *Journal of Experimental Zoology* 291, 2 (8 2001), 116–129. https://doi.org/10.1002/JEZ.1064

[49] Gregory A. Wray. 2007. The evolutionary significance of cis-regulatory mutations. *Nature Reviews Genetics 2007 8:3* 8, 3 (3 2007), 206–216. https://doi.org/10.1038/nrg2063

[50] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. https://doi.org/10.48550/ARXIV.1708.07747