Late Bloomers, First Glances, Second Chances: Exploration of the Mechanisms Behind Fitness Diversity*

Sofya Aksenyuk Szymon Bujowski Maciej Komosinski Konrad Miazga

Poznan University of Technology
Institute of Computing Science
Poznan, Poland
maciej.komosinski@cs.put.poznan.pl

Abstract

Fitness diversity is an idea in the field of evolutionary algorithms, which calls for supporting the evolution of solutions at all fitness levels simultaneously. In some cases, this idea may even extend to cultivating the worst solutions. While this may seem counterintuitive, fitness diversity has shown its promise in algorithms such as Hierarchical Fair Competition and Convection Selection. Although these algorithms share many similarities, the role fitness diversity serves in each of them is different. In Hierarchical Fair Competition, fitness diversity facilitates a constant incorporation of novel genotypes into the solutions that are already good – a mechanism we dub First Glances – and discovery of solutions through the exploration of neutral networks of different fitness levels – which we name Late Bloomers. On the other hand, Convection Selection uses fitness diversity techniques to give broken solutions time and shelter necessary to cross larger valleys in the fitness landscape – a mechanism we call Second Chances. In this work, we compare these two algorithms and their respective mechanisms over a range of numerical and 3D structure design optimization problems. We analyze the extent to which their mechanisms are utilized, and measure the impact of these mechanisms on finding good solutions.

1 Introduction

One thing can be used in several ways. For example, since mastering fire, people have used it to prepare food, provide light, heat oneself up or destroy something. We use fire not just for the sake of it, but as a tool that enables us to do other things. Similarly, in the world of evolutionary computation, one of such tools is the idea of fitness diversity. And just like fire, it should not be seen as a goal in itself, but rather as something that facilitates the inclusion of other beneficial mechanisms in the process of optimization.

Fitness diversity challenges the idea of focusing only on the good solutions. Instead, fitness diversity methods provide solutions with a chance to evolve and improve over time, no matter their quality. Perhaps the earliest representatives of fitness diversity are Fitness Uniform Selection Scheme (FUSS) [5, 6] and Fitness Uniform Deletion Scheme (FUDS) [11, 6]. In these algorithms, evolution focuses on either reproducing the solutions from the underrepresented fitness ranges (FUSS), or removing solutions from the overrepresented fitness ranges (FUDS). Another work which leads to greater diversity in the quality of solutions is Age-Layered Population Structure (ALPS) [2]. ALPS divides the population into layers based on their age, continuously filling the youngest subpopulation with new random solutions. Since the fitness of solutions is expected to increase over time, this method does also indirectly support simultaneous evolution on many fitness levels.

In this paper we want to focus on two other algorithms. First of them is Hierarchical Fair Competition (HFC) [3, 4]. HFC works similarly to ALPS, however the population is not divided

*The final version of this paper appeared in GECCO (Genetic and Evolutionary Computation Conference) Proceedings, 2024. http://dx.doi.org/10.1145/3638529.3654168.

based on the age of solutions, but on their fitness. Migration of solutions is allowed only in one direction: from worse to better subpopulations. This allows for continuous integration of novel genetic material into solutions that are already good – a mechanism we decided to call *First Glances*. Alternatively, even if the fresh genetic material does not manage to aid the optimization, the subpar solutions are free to indefinitely float along the neutral networks [13] of varied fitness levels in the fitness landscape, as they are shielded from the selective pressure which would otherwise be exerted on them by better solutions. This may allow them to find better local optima, even after the best solutions have become stuck in a subpar local optimum – therefore, they become *Late Bloomers*.

The other algorithm covered in this paper is Convection Selection (ConvSel) [9, 7]. Although Convection Selection divides the population into subpopulations based on the fitness of the solutions in a fashion similar to HFC, it allows for the migration of broken solutions from good to bad subpopulations. This way, it facilitates fixing of these broken solutions and crossing larger valleys in the fitness landscape – a mechanism we call *Second Chances*.

Although HFC and ConvSel use the mechanisms of First Glances, Second Chances and Late Bloomers to evolve solutions over many generations, it does not necessarily mean that these mechanisms actually contribute towards finding the best solutions. If a mechanism does not contribute towards finding good solutions – it may be beneficial to remove it, especially if maintaining it adds some overhead to the computations. Therefore, it is important to identify all the mechanisms present in the algorithm and measure their impact on the performance of the optimization process. Understanding them better may lead to creating new, more efficient algorithms that are composed of mechanisms that are proven to be beneficial for a given problem, or streamlining already existing algorithms by removing the mechanisms that prove to be unnecessary.

GECCO 2023 witnessed the debut of "Analysing algorithmic behaviour of optimisation heuristics" workshop. The workshop was met with a positive reception and with an anticipation of further research analyzing the inner workings of optimization heuristics [1]. In the spirit of the workshop, in this paper we set our eyes on the algorithmic behavior of HFC and ConvSel in order to find a confirmation – or a rebuttal – of the specific fitness diversity mechanisms playing a role in the process of finding good solutions. It is possible to measure a degree to which some mechanism is utilized in the process of evolution. To do this, one must analyze the evolutionary history of the best solution found during evolution. In this paper we introduce three mechanism-centric measures. We contrast their values with the quality of solutions found by HFC and ConvSel for a set of numerical benchmarks from CEC'17 and a set of evolutionary design tasks in order to determine the importance of the considered mechanisms and their actual effect on the quality of solutions found in evolution.

The structure of the paper is as follows. In Sect. 2 we describe the Hierarchical Fair Competition and Convection Selection in more detail. In Sect. 3 we discuss three mechanisms behind fitness diversity: First Glances, Second Chances and Late Bloomers. We also present the vector measures used later in order to assess the degree to which these mechanisms are actually utilized by the algorithms. In Sect. 4 we present the experimental setup. Sect. 5 shows the results of the experiments and the analysis of the role First Glances, Second Chances and Late Bloomers play in the process of finding good solutions. Finally, Sect. 6 contains a short discussion of the findings of this paper, and our plans for future research.

2 Algorithms

2.1 Hierarchical Fair Competition

Hierarchical Fair Competition (HFC) [3, 4] is a multi-population evolutionary algorithm which utilizes a hierarchical structure of the population, with M subpopulations (each containing S solutions) being assigned mutually exclusive fitness ranges. The fitness value constituting the border between two subpopulations can be seen as the "export threshold" of the worse subpopulation and – simultaneously – the "admission threshold" of the better one. The worst subpopulation has no admission threshold, and the best subpopulation has no export threshold.

Each subpopulation, with the exception of the worst one, has a buffer of candidate solutions associated with it. The subpopulations evolve independently of each other between migrations, which happen once every $M \cdot S \cdot R$ evaluations, where R is a scaling parameter controlling the frequency of migrations. During migration, solutions are moved from the admission buffers to their respective subpopulations. In the implementation used in this paper, at most $\frac{S}{2}$ best solutions are moved from the admission buffer to their respective subpopulation, filling empty slots and replacing the worst solutions in that subpopulation. The empty slots in the worst subpopulation are filled with new, randomly generated solutions. Admission buffers are emptied after each migration.

Solutions are moved from subpopulations to the admission buffers only if their fitness exceeds the export threshold of their current subpopulation. Then, such a solution is moved to the admission buffer with the highest admission threshold lower than the fitness of that solution (assuming maximization). In this paper, the synchronous version of HFC is used, where solutions are moved to the admission buffers only right before a migration.

In the original specification of HFC, the fitness ranges associated with subpopulations have to be assigned *a priori*. In this work, Adaptive Setting of Admission Thresholds (HFC-ADM) is used instead [4]. In HFC-ADM, the export threshold of the worst subpopulation is fixed, and based on the average fitness of randomly generated solutions (as measured in the calibration phase, before the proper evolution starts). All the other thresholds are recalculated before each migration. In the canonical version of HFC-ADM, the admission threshold of the best subpopulation is calculated as $f_{max} - \sigma_f$, where f_{max} is the highest fitness in the population, and σ_f is the standard deviation of fitness in the (full) population. All the other admission and export thresholds are then interpolated linearly between the export threshold of the worst subpopulation and the admission threshold of the best subpopulation. This ensures that each intermediate subpopulation covers the fitness range of equal width. However, in this paper we set the width of the best subpopulation to be the same as the width of all other subpopulations (with the exception of the worst subpopulation) to reduce the differences between the implementation of HFC and ConvSel, and therefore allow for a more fair comparison of their mechanisms (First Glances, Second Chances, and Late Bloomers).

2.2 Convection Selection

Just like HFC, Convection Selection (ConvSel) [9, 7] is a multi-population evolutionary algorithm which utilizes a hierarchical structure of the population, with M subpopulations (each containing S solutions) being assigned mutually exclusive fitness ranges. ConvSel is different from HFC in the following:

1. In the "equal width" variant of ConvSel used in this paper, the width of all fitness ranges is equal, with subpopulations covering the entire range of fitness values present in the population right before the migration.

2. There are no admission buffers.

3. Solutions migrate both upwards and downwards according to their fitness.

4. Right before a migration, fitness ranges of the subpopulations are recalculated, and during a migration, solutions are transferred to their appropriate subpopulations.

5. There is no source of entirely new, randomly generated solutions once the subpopulations have been initialized.

3 The Mechanisms Behind Fitness Diversity

3.1 Measuring the "First Glances" mechanism

First Glances is a mechanism unique to HFC. Its name refers to the ability of HFC to continuously introduce new, randomly generated solutions over the entire evolutionary run, and to incorporate their genetic material into older solutions. To see if this mechanism aids the optimization, we need to verify if the ancestry of the best solution contains solutions introduced later in the evolution, or if it consists mainly of the solutions present in the initial population.

To that end we assign a vector named *ancestor_birthtime* to every solution in the population. The length of the vector matches the number of epochs in the evolutionary run (with the n-th element of the vector corresponding to the n-th epoch, where an epoch is understood as the time between two migration events). Whenever a new solution is created randomly (be it at the start of evolution or sometime later), we assign it a vector of all 0s, except for the element with an index corresponding to the number of the current epoch, which is assigned the value of 1. When a new solution is created by mutation, it inherits the value of *ancestor_birthtime* from its parent. When a new solution is created by crossover, the values of elements of its *ancestor_birthtime* are set to the average of the values of the corresponding elements of its parents' vectors.

3.2 Measuring the "Second Chances" mechanism

3.2.1 Distribution of ancestors in the subpopulations

ConvSel lacks the mechanism of First Glances, as it does not introduce new, completely random solutions to the population during evolution. Instead, it relies on the mechanism of Second Chances. The name of Second Chances refers to the ability of ConvSel to allow broken solutions coming from good subpopulations to be maintained and fixed over time in subpar subpopulations. This way, ConvSel allows for crossing wide and deep valleys in the fitness landscape. To measure the degree to which this mechanism is actually utilized by ConvSel, we need to check the distribution of subpopulations occupied by the ancestors of the best solution. For this purpose we use a vector measure called *ancestor_spop*. The length of this vector matches the number of subpopulations in the population, with the first element of the vector corresponding to the worst subpopulation, and the last element – to the best subpopulation. At the very start of evolution, *ancestor_spop* of each solution is a vector with 0's on all positions except for the first element which is set to 1. Whenever a new solution is created by a mutation or a crossover, its value of *ancestor_spop* is calculated similarly to that of *ancestor_birthtime*.

During each migration, values of *ancestor_spop* are recalculated for all solutions in the population according to the formula:

$$ancestor\_spop = \frac{(c\_epoch - 1) \cdot ancestor\_spop + prev\_spop\_vec}{c\_epoch},$$

where *c_epoch* is the number of migrations performed so far, and *prev_spop_vec* is a vector of 0's with 1 in the position representing the subpopulation occupied by the solution before the migration. This way, the elements of *ancestor_spop* should reflect the fraction of the cumulative evolutionary time the ancestors of a solution spent in specific subpopulations over the entire evolutionary process.

3.2.2 Vertical mobility of the ancestors

Although *ancestor_spop* can tell us if the ancestors of the best solution spent a lot of time in low-quality subpopulations, it cannot reliably detect if they regularly visited low-quality populations for a short time. To cover this possible scenario of utilizing the Second Chances mechanism, we define an additional vector measure – *ancestor_mobility*. The length of this vector is equal to $number\_of\_populations \cdot 2 + 1$. Each element of the vector corresponds to a specific size and direction of jumps between the subpopulations taken by the ancestors of a solution during migrations. The first element of the vector corresponds to the migration from the best subpopulation to the worst one, the middle element corresponds to a jump of size 0 (i.e., a solution not changing its subpopulation during migration), and the last element – a jump from the worst subpopulation to the best one. Initially, *ancestor_mobility* is set to all 0's except for the position corresponding to no movement, which is set to 1. Whenever a new solution is created by a mutation or a crossover, its value of *ancestor_mobility* is calculated similarly to that of *ancestor_birthtime* and *ancestor_spop*.

During each migration, values of *ancestor_mobility* are recalculated for all solutions in the population according to the formula:

$$ancestor\_mobility = \frac{(c\_epoch - 1) \cdot ancestor\_mobility + jump\_vec}{c\_epoch},$$

where c_epoch is the number of migrations performed so far, and $jump\_vec$ is a vector of 0's with 1 in the position representing the size and direction of a jump between the subpopulations made by this solution during the current migration.

3.3 Measuring the "Late Bloomers" mechanism

Ancestor_spop can also be calculated for HFC, although in the context of this algorithm its interpretation becomes slightly different. Since HFC does not allow for Second Chances, ancestors occupying subpar subpopulations cannot be linked to fixing broken solutions. Instead, this measure allows one to observe the presence of "Late Bloomers", i.e., solutions that existed and drifted for a long time at lower fitness levels, perhaps finding novel promising areas of the fitness landscape.

We do not expect Late Bloomers to be present in ConvSel, as the downward migration will inevitably introduce fragments of good solutions to low-quality subpopulations, changing their role in the search process. Similarly, the presence of "First Glances" might introduce novel genetic material to low-quality subpopulations, so we can only confidently confirm the presence of Late Bloomers in the absence of First Glances.

4 Methods

4.1 Mathematical benchmarks

Test functions used in this work are based on CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization [14]. All test functions are minimized and the search range is $[-100, 100]^D$, where D is the dimensionality of the problem. In this work, we use $D = 30$. The experiments use the Python implementation of the functions [12]. For the experiments we selected the following 19 functions: $F1$, $F3$–$F10$, and $F21$–$F30$, as defined in [12].

4.2 Evolutionary design benchmarks

In addition to numerical benchmarks, in this investigation we include two optimization problems of a different nature. They concern evolutionary design, i.e., evolving 3D structures or agents. In these problems, the search space is discrete-continuous, and due to a complex genotype-to-phenotype mapping, the fitness landscape is highly rugged and chaotic. For these optimization problems we use Framsticks simulation environment [8, 10] and an open-source Python interface that allows running customized evolutionary algorithms.

Framsticks simulates and evolves three-dimensional designs controlled by recurrent neural networks and is available as a native application for all major operating systems. In the experiments, a simple MechaStick physics engine is used to simulate elastic "sticks" – bodies are composed of points with a mass that are connected with joints. Control systems of simulated agents are composed of artificial neurons of various types, including sensors and effectors, that can be freely connected – with recurrent and parallel connections allowed. In the experiments reported here, the following neuron types are used: a sigmoid neuron, a neuron that always outputs a constant value of 1, two kinds of effectors (rotation around two axes), and two kinds of sensors (tilt and touch).

A body and a control system are described by a genotype that can undergo mutation, crossover, and repair. There are many genetic encodings in Framsticks, each with its own dedicated genetic operators. The genetic encoding used here is the default one, denoted as *f1*. This is a high-level encoding that directly maps each gene into a single part of the body or control system. For example, the "X" symbol creates a "stick" in the body, parentheses "(" and ")" make a branching in the body so that tree-like structures can be represented, and commas "," separate individual branches. Additional symbols (genes) influence the length of sticks, the plane of branching, or physical properties like the coefficient of friction. Neurons are encoded in square brackets, with individual connections separated by commas. Each connection is described by the relative index of the input neuron and the value of the connection weight. Mutation operators modify specific aspects of the genotype – for example adding or removing an "X", adding or removing a branching "(...)", adding or removing a neural connection, changing connection weight, etc. The crossover operator exchanges two randomly selected substrings in parent genotypes. The simplest genotype in *f1* is **X** (a single "stick"). More complex examples are **XX(X,X(X,X,,,))** and **X[N,1:0.3](X,X[N,−1:0.2,1:0.4][N,−2:0.8])**.

The two optimization functions included in this investigation are vertical position of the center of mass of a passive structure (neural network is disabled) and velocity on land (neural network is coevolved along with the body). The functions are further denoted as "vertpos" and "velocity", respectively, and both functions are maximized.

4.3 Experimental setup

The goal of the experiments is not to find the best performing parameters for the considered algorithms, nor is it to directly compare the quality of their results. The goal of the experiments is to analyze the influence of the mechanisms used by HFC and ConvSel on the fitness of the best solutions found during the evolutionary process. For this reason, the parameter values used in the experiment were selected as the parameters that have shown good performance in the past, but they have not been tuned specifically for the test functions used in this work. Due to the similarity of the general structure of HFC and ConvSel, the same values of parameters were used for both algorithms.

Solutions in the numerical experiments are represented by vectors of floating point numbers. Mutation consists in adding a random vector, with value at each position drawn at random from the normal distribution $\mathcal{N}(0, 0.2)$. If any element of the vector after mutation exceeds the allowed range, this value is reflected back towards the allowed range. Crossover is performed as a weighted average of two parent vectors with weights r and $1-r$, where r is drawn on each mutation from the uniform distribution $\mathcal{U}(0, 1)$.

Both HFC and ConvSel use a generational EA within their subpopulations, with each new subpopulation consisting of 80% mutated solutions, and the remaining 20% are crossed-over solutions.

The initial population for both algorithms is filled with randomly generated solutions. Generation of random solutions for both the initial population and the First Glances mechanism is done in the same way. For numerical benchmarks, a random solution is a random vector drawn from the uniform distribution over the allowed range in each dimension. For evolutionary design problems, a random solution is created by sequentially mutating the simplest structure until it becomes complex enough (the number of body parts ranges from 2 to 15, and the number of neurons – from 1 to 15).

The parameters of the algorithms used in the experiments are as follows: $migration\_interval = 10$, $number\_of\_populations = 25$, $subpop\_size = 50$, $t\_size \in \{2, 5\}$.

The total number of evolutionary runs for the mathematical benchmarks was 2 (algorithms) × 2 (tournament sizes) × 19 (number of benchmark functions) × 50 (independent repetitions) = 3 800.

The total number of evolutionary runs for the evolutionary design benchmarks was 2 (algorithms) × 2 (tournament sizes) × 2 (number of benchmark functions) × 30 (independent repetitions) = 240.

Each run consisted of 300 generations and required 300 (generations) × 25 (number of populations) × 50 (size of each population) = 375 000 function evaluations.

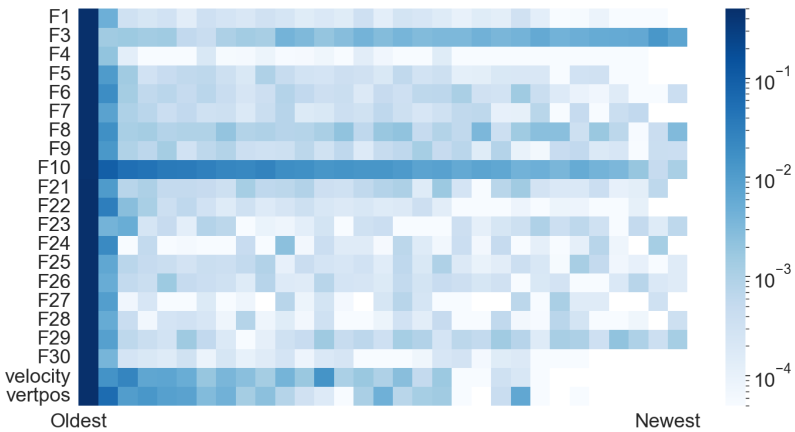
5 Results

5.1 Evaluating the "First Glances" mechanism

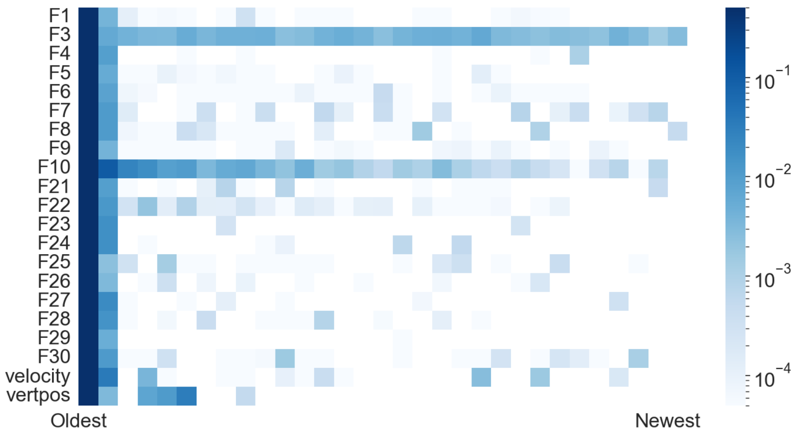
To evaluate the degree of utilization of First Glances mechanism in HFC, we analyze the values of *ancestor_birthtime* vectors taken from the best solutions found during evolution, averaged over multiple runs. The averaged values of the *ancestor_birthtime*, as calculated for different optimization problems, are shown in Fig. 1.

The heatmaps clearly show that the vast majority of the genetic material which ends up in the final, best solution, originates from the initial population. It does not invalidate the mechanism of First Glances, as the subsequent epochs will naturally introduce a far lower number of new random solutions than was created initially. The second epoch is also the source of a decent amount of genetic material, however for most problems, the role of First Glances in the following epochs is diminished. Despite that, new random solutions still manage to sometimes influence the final solution. This effect is, however, much more pronounced in the case of a lower selective pressure ($t\_size = 2$, Fig. 1a) than higher selective pressure ($t\_size = 5$, Fig. 1b), where for most cases the novel solutions introduced in the later epochs almost never influence the final solution.

Among the tested benchmarks, the behavior of a few stands out. Numerical functions $F3$ and $F10$ utilize the mechanism of First Glances far more than the other functions. Function $F3$ (Shifted



(a) $t\_size = 2$



(b) $t\_size = 5$

Figure 1: The values of *ancestor_birthtime* vectors of the best solutions found by the HFC algorithm, averaged from independent evolutionary runs. Optimization problems shown in rows. The color scale is logarithmic.

and Rotated Zakharov Function) uses progressively more First Glances as the evolution progresses, which can be explained by the fact that this function is fairly flat in the middle of the search range, with just one local optimum, so additional First Glances are likely to introduce useful solutions, as the basin of the global optimum is very wide and the convergence to the global optimum might be slow. On the other hand, function $F10$ (Shifted and Rotated Schwefel's Function) uses First Glances less as the evolutionary time progresses, as it is full of deep local optima, so while initially First Glances may aid in finding basins of better local optima, once the population starts to converge to a specific local optimum, it becomes harder to find better solutions.

Evolutionary design benchmarks (with the exception of velocity for $t\_size = 5$) appear to utilize First Glances more than most of the mathematical benchmarks.

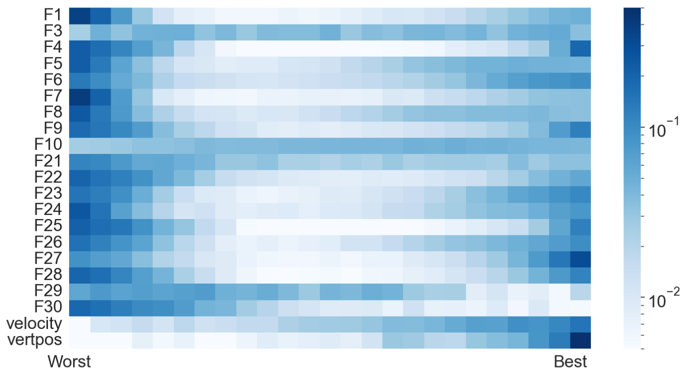
Overall, while the mechanism of First Glances is used by HFC, for most problems its importance quickly decreases after the first epochs. Similarly, its importance appears lower when the selective pressure is higher – possibly because this makes it harder for new solutions to compete against other solutions, which are already at least partially optimized.

5.2 Evaluating the "Second Chances" mechanism

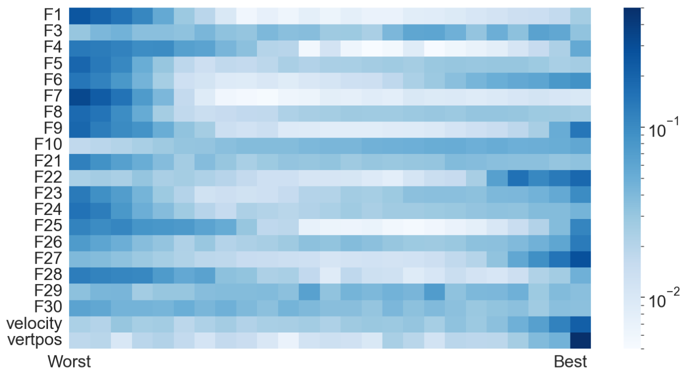
5.2.1 Distribution of ancestors in the subpopulations

The heatmap for *ancestor_spop*, created in a similar fashion as the one for *ancestor_birthtime*, is shown in Fig. 2.

For most mathematical benchmarks, the ancestors of the best solution tend to occupy the best and the worst subpopulations, mostly avoiding the subpopulations of a middling quality. This could



(a) $t\_size = 2$



(b) $t\_size = 5$

Figure 2: The values of *ancestor_spop* vectors of the best solutions found by the ConvSel algorithm, averaged from independent evolutionary runs. Optimization problems shown in rows. The color scale is logarithmic.

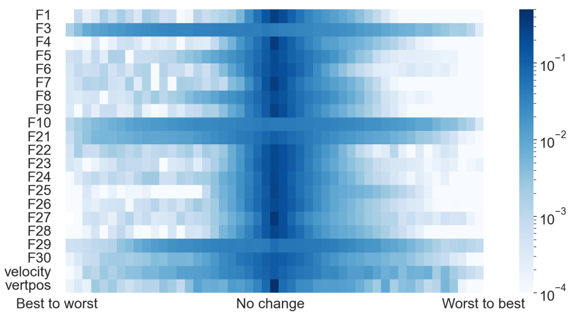
suggest that ConvSel actually gives broken solutions Second Chances, although the low utilization of the subpopulation of middling quality suggests that fixing the broken solutions in these problems does not require incremental improvements, but rather just one or two lucky mutations. Again, the behavior of a few mathematical benchmarks stands out. For $F3$ and $F10$, all subpopulations are utilized to a similar degree. For $F21$, $F29$, and $F30$, all of which are functions composed of several other, simpler functions, the worst subpopulations are utilized the most, with ancestors of the best solutions found in $F30$ spending almost no time in the best subpopulation.

The behavior of ConvSel is visibly different for the tasks of evolutionary design. For both of the tasks there is a general tendency of better subpopulations being utilized more, with the worst subpopualtions being heavily underutilized. It may suggest that in the tasks of evolutionary design, the fitness of broken solutions is closer to that of a parent solutions, unlike the mathematical benchmarks where mutation can often change the quality of a solution to a very high degree. It may also be related to the fact that for these tasks the population did not manage to converge before the end of the run, so the data we see captures the phase in which it is still easy to improve the best solutions.

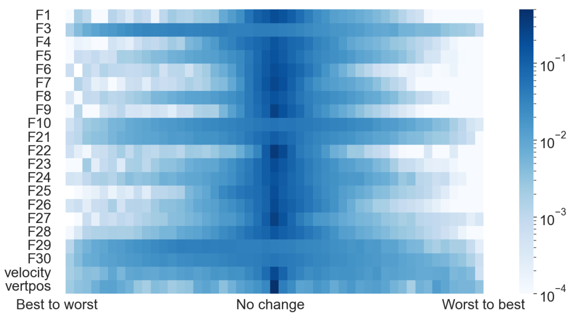
For both mathematical and evolutionary design benchmarks, increased selective pressure (Fig. 2b) causes the middling subpopulations to be utilized more.

5.2.2 Vertical mobility of the ancestors

The averaged values of *ancestor_mobility* vectors are shown in Fig. 3. There is a clear inverse relationship between the size of a jump between subpopulations and the frequency of its occurrence, with the ancestors of the best solutions not changing their subpopulations during most migrations. While big improvements almost never happen, and improvements are usually small and gradual, the degree to which a solution might get broken is more varied, as indicated by steeper, yet longer tails on the left side of Fig. 3.



(a) $t\_size = 2$



(b) $t\_size = 5$

Figure 3: The values of *ancestor_mobility* vectors of the best solutions found by the ConvSel algorithm, averaged from independent evolutionary runs. Optimization problems shown in rows. The color scale is logarithmic.

Once more, mathematical benchmarks $F3$, $F10$, $F21$, $F29$ and $F30$ stand out. Their distribution of sizes of a jump between subpopulations is fairly uniform, most likely indicating a very low range of fitness values in the full population, or a very rugged fitness landscape.

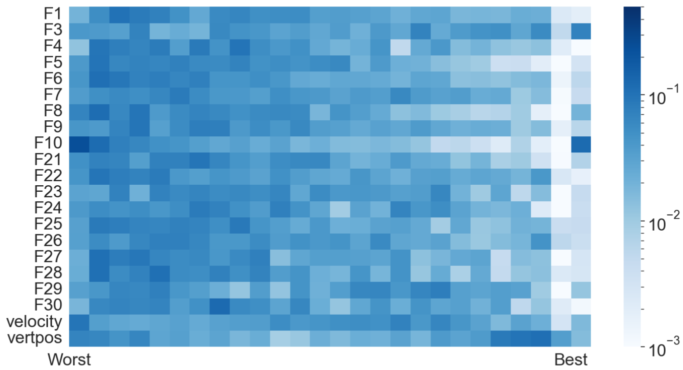
Although the tails of *ancestor_mobility* are long for the *vertpos* benchmark, this function has the highest probability of the ancestors not changing the subpopulation during migrations, which may again suggest that the population is still quickly improving within the best subpopulation.

In the presence of a stronger selective pressure (Fig. 3b), the values of *ancestor_mobility* are more spread out, which may indicate both a greater mobility of solutions and a smaller range of fitness values in the full population, which pushes the fitness ranges of the subpopulations closer together.

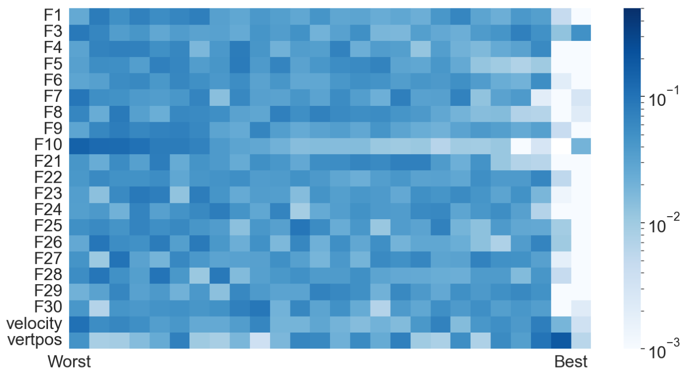
Overall, the results confirm that the mechanism of Second Chances is utilized by ConvSel. We have found a proof of both high and low quality subpopulations being used by the ancestors of the best solutions. We have also detected in the ancestry of the best solutions a presence of solutions breaking and then being slowly fixed over time.

5.3 Evaluating the "Late Bloomers" mechanism

Figure 4 presents the values of *ancestor_spop* vectors of the best solutions found by the HFC algorithm in different optimization problems. Surprisingly, Fig. 4 reveals that most of the ancestors come from low quality subpopulations, with very short time spent by them in the best subpopulation. This might be explained by the structure of a population in HFC, which allows solutions to move upwards, but not downwards. The solutions reaching the best subpopulation become,



(a) $t\_size = 2$



(b) $t\_size = 5$

Figure 4: The values of *ancestor_spop* vectors of the best solutions found by the HFC algorithm, averaged from independent evolutionary runs. Optimization problems shown in rows. The color scale is logarithmic.

therefore, trapped there, with the worst of them being regularly replaced by solutions rising from the lower subpopulations. In effect, solutions in a specific subpopulation can only have ancestors from subpopulations not better than that subpopulation, which eventually leads to the lower quality subpopulations being overrepresented in the ancestry of solutions.

Increasing the selective pressure ($t\_size = 5$, Fig. 4b) causes the ancestors of the best solution to spend more time in high quality subpopulations, making the distribution more even than it is for a lower selective pressure ($t\_size = 2$, Fig. 4a). Intriguingly, it simultaneously appears to decrease the time the ancestors of the best solution spent in the best subpopulation.

While in theory the best solutions could survive over longer periods of evolutionary time in the best subpopulation, avoiding the Late Bloomers mechanism, Fig. 4 clearly shows that this is not the case. A possible exception to that rule can be observed for the *vertpos* benchmark, behavior of which might, however, be explained by the population not being yet converged. Compared to the mechanism of First Glances, the mechanism of Late Bloomers seems to be more robust and utilized more during evolution.

5.4 The relationship between the utilization of the mechanisms and the quality of the solutions

To determine the degree to which different mechanisms were utilized by evolution, we introduce several scalar measures based on the vector measures described in Sect. 3. Measures using suffix *cog* (shorthand for "center of gravity") are defined as a weighted sum of indices of a vector, with the elements of the vector serving as weights, divided by the length of the vector. Measures using suffix *std* are defined as the standard deviation of the values of the vector. We include the following measures in our analysis: *birthtime_cog* (for HFC), *mobility_std* (for ConvSel), *spop_cog*, and *spop_std* (for HFC and ConvSel).

We have analyzed the relationships between the utilization of different mechanisms and the fitness of the best solutions for each combination of the fitness function and the size of the tournament. To express these relationships quantitatively, we calculated Spearman's rank correlation coefficients. The decision to use Spearman's correlation was dictated by the observation that the distributions of fitness values and the utilization of mechanisms are not always normal – in some cases, clusters and outliers were detected.

	HFC			ConvSel		
	FG	LB		SC		
	birthtime_cog	spop_cog	spop_std	mobility_std	spop_cog	spop_std
F1	0.09	-0.04	-0.07	0.18	0.11	0.14
F3	**0.21**	-0.14	**-0.34**	-0.19	0.02	-0.12
F4	0.14	-0.15	-0.06	0.17	0.14	-0.06
F5	-0.06	-0.05	-0.02	-0.11	0.03	-0.21
F6	0.10	**0.21**	-0.18	**-0.30**	0.16	-0.20
F7	-0.01	-0.05	-0.05	0.01	-0.09	0.10
F8	-0.11	0.07	-0.05	-0.06	-0.04	-0.07
F9	0.17	0.05	**-0.32**	0.22	-0.04	0.12
F10	-0.14	0.06	**-0.25**	0.03	-0.03	-0.02
F21	0.16	0.00	-0.04	**-0.72**	-0.03	**-0.36**
F22	**0.24**	0.11	**-0.41**	0.20	0.09	0.15
F23	0.09	0.15	-0.14	0.16	0.01	0.09
F24	0.04	0.08	-0.05	-0.07	**-0.56**	0.25
F25	0.17	0.16	**-0.27**	0.04	**0.31**	**-0.41**
F26	0.08	-0.02	-0.14	-0.25	**0.47**	**-0.45**
F27	0.11	-0.05	-0.10	0.14	0.13	0.10
F28	0.14	-0.08	-0.06	0.23	0.16	-0.08
F29	0.14	0.01	-0.01	**-0.29**	0.25	-0.10
F30	-0.01	0.14	-0.04	0.06	0.09	0.01
velocity	-0.17	0.10	0.07	0.11	0.26	-0.10
vertpos	-0.10	0.13	-0.08	0.16	0.06	0.20

(a) $t\_size = 2$

	HFC			ConvSel		
	FG	LB		SC		
	birthtime_cog	spop_cog	spop_std	mobility_std	spop_cog	spop_std
F1	0.16	0.16	**-0.45**	0.06	0.12	0.09
F3	0.07	0.01	**-0.24**	**-0.42**	-0.04	**-0.52**
F4	0.11	0.02	**-0.23**	0.20	0.05	0.27
F5	0.08	0.14	-0.15	0.17	-0.09	0.11
F6	**0.26**	**0.27**	**-0.25**	-0.14	0.23	-0.13
F7	0.06	0.07	-0.05	**0.37**	-0.01	0.04
F8	0.16	-0.01	-0.11	0.02	0.03	0.03
F9	**0.21**	0.14	**-0.48**	-0.14	-0.04	-0.03
F10	**-0.23**	0.07	0.02	**0.31**	**0.34**	**0.32**
F21	0.05	0.09	-0.03	**-0.80**	**0.81**	**-0.71**
F22	**0.31**	0.19	**-0.22**	0.03	0.00	0.04
F23	0.08	0.11	**-0.26**	**0.69**	**0.62**	**0.45**
F24	-0.13	-0.13	0.02	-0.08	0.01	-0.03
F25	0.16	-0.07	-0.03	0.06	**0.28**	-0.02
F26	**0.23**	-0.06	**-0.28**	0.05	0.19	-0.12
F27	-0.01	0.13	-0.19	-0.12	-0.10	-0.05
F28	0.17	0.14	-0.19	**-0.36**	0.13	-0.12
F29	0.06	0.12	-0.17	-0.22	**0.29**	**-0.31**
F30	0.13	0.04	-0.16	0.17	-0.23	-0.09
velocity	0.07	-0.08	0.16	**0.36**	0.34	0.31
vertpos	0.03	0.15	0.23	0.15	-0.01	0.22

(b) $t\_size = 5$

Table 1: Spearman correlation coefficients between the utilization of different mechanisms (FG – First Glances, LB – Late Bloomers, SC – Second Chances) represented as a weighted sum of indices (*cog*) or standard deviation (*std*) of values of vector measures, and fitness of the best solutions found by the algorithms for different test functions. Statistically significant correlations ($p = 0.05$) are shown in bold.

The values of the correlations are presented in Tables 1a and 1b. Statistically significant correlations ($p = 0.05$) are shown in bold. The relationship between the value of a measure and the fitness of a solution (to which we will refer as a "utility" of a measure) is problem-dependent, with some of the test functions benefiting more from certain mechanisms than others.

First, we analyze the mechanisms of HFC: First Glances and Late Bloomers. For mathematical benchmarks, there is a general – yet not unanimous – preference of utilization of the mechanism of First Glances, as indicated by positive correlations with fitness for *birthtime_cog*. This suggests that the mechanism of First Glances aids evolution. For mathematical benchmarks, it appears overall beneficial for solutions to utilize all subpopulations to a similar degree (as shown by the negative utility of *spop_std*), avoiding the overrepresentation of low-quality subpopulations, or even to focus on the subpopulations of higher quality (as shown by the positive utility of *spop_cog*). This suggests that the mechanism of Late Bloomers may not always be beneficial, especially when the influx of solutions from the low-quality subpopulations is too strong, which may disrupt the exploration of neutral networks of higher fitness. This negative aspect of Late Bloomers could potentially be mitigated by adding a mechanism of genotypic diversity into the subpopulations of HFC.

For the problems of evolutionary design, the utilities of First Glances and Late Bloomers often do not match the trends seen for the mathematical benchmarks. However, for these problems, none of the correlations related to HFC were statistically significant.

When it comes to ConvSel, the utility of the mechanism of Second Chances appears to be far more problem-dependent that it is for First Glances and Late Bloomers. Statistically significant correlations can be found in both directions. This suggests that while the absolute value of the utilities is often quite high (for example for $F21$ or $F23$) – which means that the mechanism of Second Chances plays a significant role in evolution – the exact role it plays depends on the problem itself.

The sign of utility for a mechanism changes the interpretation of the correlation. Values close to zero indicate that the utilization of the mechanism did not noticeably influence the search process. Just because a mechanism is utilized more when the final fitness is worse does not necessarily mean that the mechanism is detrimental. Instead it may suggest that when it is hard to discover good solutions during evolution, evolution relies more than usually on that mechanism in order to overcome the issue. The positive correlation, on the other hand, should indicate that the mechanism actively aided the search not by mitigating the losses, but rather by amplifying the gains.

In most cases, increased selective pressure heightens the role the mechanisms play, with the utilities becoming more pronounced.

6 Summary

In this paper we took a closer look at the mechanisms present in two fitness diversity algorithms – Hierarchical Fair Competition (HFC) and Convection Selection (ConvSel). We identified three mechanisms enabled by supporting simultaneous evolution on several fitness levels: First Glances (continuous introduction of novel solutions to the population), Second Chances (fixing broken solutions which allows for crossing fitness valleys), and Late Bloomers (finding novel promising areas in the fitness landscape by drifting along some specific fitness level). We also developed measures which allow one to estimate the level of utilization of these mechanisms.

The main contribution of this paper is an in-depth exploration and analysis of different mechanisms behind fitness diversity. Although the utilization of First Glances is relatively low – the vast majority of the genetic material which ends up in the final, best solution, originates from the initial population – this mechanism seems to generally benefit the quality of solutions found by HFC. On the other hand, Late Bloomers, while utilized more due to the structure of the algorithm, may in some cases allow low-quality subpopulations to impede the efforts of higher-quality subpopulations. Although it is clear that the mechanism of Second Chances plays an important role in the behavior of ConvSel, the exact way it aids optimization depends strongly on the problem itself.

While the findings of our experiments are already useful, we acknowledge the limited scope of the research presented in this paper. The importance of different mechanisms will differ depending on the parameter values of the algorithm, while in the experiments discussed above most parameters were fixed – both these common to all evolutionary algorithms, and these specific to HFC and ConvSel. Additionally, more focus should be put on exploring how the importance of different mechanisms changes depending on the length of the evolutionary process. After the population converges, the overall behavior of the algorithm changes as well, which should be taken into consideration when examining the utility of different mechanisms.

Despite these shortcomings – which we intend to tackle in further research – we believe that this investigation constitutes an important first step towards a better understanding not only *if* the fitness diversity algorithms work well, but also *how* and *when* they work. It shows the potential hidden in analyzing the relations between the optimization problems and the mechanisms employed by the optimization algorithms.

In the future, we also want to compare the properties of optimization problems [15] with the utility of different mechanisms, which should help one discover dependencies that will guide researchers towards developing new algorithms dedicated for problems with specific characteristics.

Acknowledgements

MK and KM acknowledge the support by the Polish Ministry of Education and Science, grant no. 0311/SBAD/0740. SA and SB acknowledge the support by the Faculty of Computing, Poznan University of Technology, through the funds provided by the Polish Ministry of Education and Science, grant no. 0311/SBAD/0732. KM acknowledges the support by the Faculty of Computing, Poznan University of Technology, through the funds provided by the Polish Ministry of Education and Science, grant no. 0311/SBAD/0744.

References

[1] Lauren Hayward and Andries Engelbrecht. How to tell a fish from a bee: Constructing metaheuristic search behaviour characteristics. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, GECCO '23 Companion, page 1562–1569, New York, NY, USA, 2023. Association for Computing Machinery. `doi:10.1145/3583133.3596338`.

[2] Gregory S. Hornby. Alps: the age-layered population structure for reducing the problem of premature convergence. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 815–822, 2006.

[3] Jian Jun Hu and Erik D. Goodman. The hierarchical fair competition (HFC) model for parallel evolutionary algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02*, volume 1, pages 49–54. IEEE, 2002.

[4] Jianjun Hu, Erik Goodman, Kisung Seo, Zhun Fan, and Rondal Rosenberg. The hierarchical fair competition (HFC) framework for sustainable evolutionary algorithms. *Evolutionary Computation*, 13(2):241–277, 2005.

[5] Marcus Hutter. Fitness uniform selection to preserve genetic diversity. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02*, volume 1, pages 783–788. IEEE, 2002.

[6] Marcus Hutter and Shane Legg. Fitness uniform optimization. *IEEE Transactions on Evolutionary Computation*, 10(5):568–589, 2006.

[7] Maciej Komosinski and Konrad Miazga. Tournament-based convection selection in evolutionary algorithms. In *Parallel Processing and Applied Mathematics: 12th International Conference, PPAM 2017, Lublin, Poland*, pages 466–475. Springer, 2018.

[8] Maciej Komosinski and Szymon Ulatowski. Framsticks: Creating and understanding complexity of life. In Maciej Komosinski and Andrew Adamatzky, editors, *Artificial Life Models in Software*, chapter 5, pages 107–148. Springer, London, 2nd edition, 2009. URL: `https://www.springer.com/978-1-84882-284-9`.

[9] Maciej Komosinski and Szymon Ulatowski. Multithreaded computing in evolutionary design and in artificial life simulations. *The Journal of Supercomputing*, 73(5):2214–2228, 2017. `doi:10.1007/s11227-016-1923-4`.

[10] Maciej Komosinski and Szymon Ulatowski. Framsticks website, 2025. URL: `http://www.framsticks.com`.

[11] Shane Legg and M. Hutter. Fitness uniform deletion: a simple way to preserve diversity. In *Proc. of the 7th annual conf. on Genetic and evolutionary computation*, pages 1271–1278, 2005.

[12] Duncan Tilley and Łukasz Wolanin. Python module for CEC 2017 single-objective optimization test function suite. `https://github.com/tilleyd/cec2017-py`, 2022.

[13] Claus O. Wilke. Adaptive evolution on neutral networks. *Bulletin of mathematical biology*, 63(4):715–730, 2001.

[14] Guohua Wu, Rammohan Mallipeddi, and Ponnuthurai Nagaratnam Suganthan. Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization. *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report*, 2017.

[15] Feng Zou, Debao Chen, Hui Liu, Siyu Cao, Xuying Ji, and Yan Zhang. A survey of fitness landscape analysis for optimization. *Neurocomputing*, 503:129–139, 2022.