Revealing the Inner Dynamics of Evolutionary Algorithms with Convection Selection*

Maciej Komosinski Konrad Miazga

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

Abstract

Evolutionary algorithms are stochastic algorithms so they tend to find different solutions when run repeatedly. However, it is not just the solutions that vary – the very dynamics of the search that led to finding these solutions are likely to differ as well. It is especially in the algorithms with complex population structures – such as convection selection where a population is divided into subpopulations according to fitness values – where an opportunity for highly diverse dynamics arises. This work investigates the way evolutionary dynamics of subpopulations influence the performance of evolutionary algorithms with convection selection. We employ a demanding task of evolutionary design of 3D structures to analyze the relation between the properties of the optimization task and the features of the evolutionary process. Based on this analysis, we identify the mechanisms that influence the performance of convection selection, and suggest ways to improve this selection scheme.

1 Introduction

It is easy to modify the behavior of an evolutionary algorithm by changing the values of its parameters. However, although one can control the logic of the algorithm and its parameter values when running an experiment, they are not the only aspects that influence the performance of the algorithm. Since the course of an evolutionary algorithm usually differs between runs – even for the same parameter values – when looking for sources of variability of the results, one should also consider the inner dynamics resulting from the interaction of the algorithm and the fitness landscape.

For the purpose of this paper, by inner evolutionary dynamics, we understand all kinds of statistically measurable processes acting upon the state of the population throughout the evolution. The influence of inner evolutionary dynamics can be either beneficial (e.g., neutral genetic drift [8] or population diversity [7]) or detrimental (e.g., premature convergence [6]). The presence of these dynamics usually depends on random events during evolution (such as the complete removal of some phenotypic alleles from the population, or the accumulation of mutations in a genotype allowing for traversing a valley in the fitness landscape), and therefore cannot be controlled directly with the parameters of the algorithm.

In this work, we show that the examination of the influence of the inner dynamics of an algorithm on the performance of the search process can provide useful insights into the behavior of the algorithm and its interaction with specific optimization problems. Although our experiments are based on convection selection [4, 2], the method proposed in this work can also be applied to other algorithms.

*The final version of this paper appeared in GECCO (Genetic and Evolutionary Computation Conference) 2023 Companion. http://dx.doi.org/10.1145/3583133.3590708.

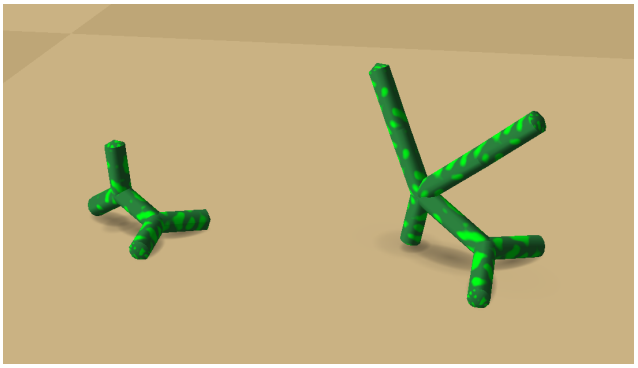


Figure 1: Simple examples of simulated structures encoded by *f1* genotypes. Left: `X(X,RRX(X,X))`. Right: `LLRXR(X,LLRXR(X,,XX,LLXX))`.

2 Methods

2.1 Algorithm

For the purpose of the experiments reported in this paper, convection selection was used as an exemplary algorithm because of its relatively complex population structure, which allows for the creation of more interesting features of behavior of the population during evolution. In convection selection [4, 2], solutions are assigned to subpopulations and these subpopulations are ordered according to fitness. Each subpopulation processes solutions of a limited fitness range – the fitness range of all individuals is split into disjoint ranges assigned to each subpopulation. All subpopulations evolve independently with occasional migrations, during which fitness ranges of subpopulations are recalculated, and all solutions are reassigned to subpopulations based on their fitness. This selection mechanism influences the migration policy and acts as an equalizer of selective pressure, because it allows individuals to only compete with other individuals of similar fitness range.

2.2 Fitness functions

The computational experiments reported in this work concern evolutionary design of three-dimensional structures. The experiments have been performed using the Framsticks environment [3, 5].

Three-dimensional structures are encoded in genotypes and are subject to mutation, crossover and repair operators. There are a number of genetic encodings in Framsticks [5], and here we use two of them denoted by *f1* and *f9*.

In the *f1* direct encoding, the structure is represented by a string of symbols: "X" denotes a stick, parentheses "(" and ")" denote branching with commas "," separating individual branches, and additional letters modify the structure ("L" to lengthen sticks, "R" to rotate the plane of branching by 45°, etc.). Fig. 1 demonstrates two sample structures. The mutation operator in *f1* modifies individual aspects of the genotype, e.g. adds and removes "X", parentheses, commas, etc. Crossover swaps random substrings in parent genotypes, and the repair operator which is always run after mutation and crossover tries to fix potentially uneven parentheses. Evolution using this encoding is subject to several constraints: the minimal and maximal stick length is limited, and the syntax does not allow some of the morphologies, e.g., cycles (loops) cannot be formed.

The genotype in the *f9* encoding consists of only six letters: D, U, L, R, B, F, which correspond to six directions in the 3D space (down, up, left, right, back, forth). The interpretation of *f9* genotypes resembles drawing a path in 3D "turtle graphics". Compared to *f1*, this encoding is limited – the length of the turtle's step is fixed, and only 90° turns are allowed, as demonstrated in Fig. 2. However, cycles can be formed, which is not possible in *f1*. The mutation operator in *f9* replaces 10% of randomly selected letters with random letters. Single-point crossover is used.

The fitness function used in the experiments is the vertical position of the center of mass of a structure. During the evaluation, the structure is first simulated for some time in order for it

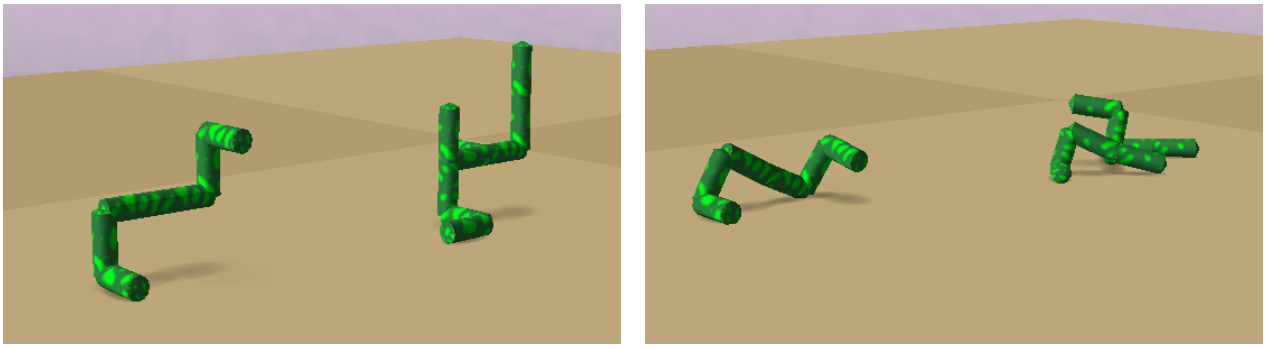


Figure 2: Simple examples of structures encoded by *f9* genotypes: `LUFFUR` and `DURBUDDRBFLUFBFFUU`. Left picture: initial state when starting the simulation of both structures. Right picture: after the initial stabilization period.

to stabilize (the displacement of the center of mass in 100 consecutive simulation steps should fall below 0.01). Then, the structure is simulated for 500 steps, every 100 steps the vertical position of the center of mass is recorded, and the final fitness of the structure is the average of the recorded values.

2.3 Experimental setup

In the experiments, we use convection selection in a steady state evolutionary algorithm (replacing one solution at a time) with random deletion of solutions to keep population size constant. The width of a fitness range for each subpopulation is equal, i.e., the full range of best-to-worst fitness is divided into M equal intervals, where M is the number of subpopulations.

Probabilities of applying the mutation and crossover operators are both set to 50%. Elitism is employed to ensure that the best solution from each subpopulation will not be deleted, so the quality of the best solution in the full population increases monotonically. The parameter values of convection selection which have been shown to perform well for similar evolutionary design tasks were selected, in accordance with the results of existing research [2]:

- Number of subpopulations $M = 25$
- Migration period multiplier $R = 10$
- Tournament size $k = 5$
- Subpopulation size $S = 50$

Each experiment evaluates 500 000 solutions in total.

We performed two sets of experiments, one for the *f1* genetic encoding (**height_f1**) and one for *f9* (**height_f9**). For every combination of tasks and parameter values, 100 independent evolutionary runs were conducted.

3 Features of evolutionary dynamics

Statistical features proposed in this section will be used later in Sect. 4 as descriptors of evolutionary dynamics, with each feature summarizing one aspect of the entire evolutionary run. When calculating the values of the features the worst subpopulation is assigned index 0, the second worst is assigned index 1, and so on.

3.1 Features based on the improvements of the best solution

Features in this group are based on the information about all the *improvements* that occurred during evolution, i.e., all solutions that at any point during the evolution held the title of "the best solution found so far".

- *b_imp_count* – the number of improvements throughout the entire evolutionary run.
- *b_in_best_since* – the number of steps (counting from the end of the evolutionary run) since the last improvement originating from outside of the best subpopulation.
- *b_imp_xov_perc* – the percentage of improvements being a direct result of a crossover.
- *b_imp_subpop_mean* – the average index of the subpopulation from which the improvements originated.
- *b_imp_subpop_stdev* – the standard deviation of indexes of subpopulations from which the improvements originated.
- *b_imp_fitdelta_stdev* – the standard deviation of the average size of the improvement.
- *b_imp_waittime_stdev* – the standard deviation of lengths of time intervals between two consecutive improvements.

3.2 Features based on the dissimilarity of solutions

Features in this group are based on the genetic dissimilarity between solutions found in the final population (at the end of the final cycle of independent evolution). The dissimilarity between the solutions is calculated using the Levenshtein distance [1].

Let $dissim(P_x, P_y) = avg(\{lev(s_a, s_b) : s_a \in P_x, s_b \in P_y\}$. Then:

- $d\_dissim\_inter\_mean = avg(\{dissim(P_x, P_y) : P_x, P_y \in subpops \wedge P_x \neq P_y\})$
- $d\_dissim\_inter\_stdev = stdev(\{dissim(P_x, P_y) : P_x, P_y \in subpops \wedge P_x \neq P_y\})$
- $d\_dissim\_intra\_mean = avg(\{dissim(P, P) : P \in subpops\})$
- $d\_dissim\_intra\_stdev = stdev(\{dissim(P, P) : P \in subpops\})$

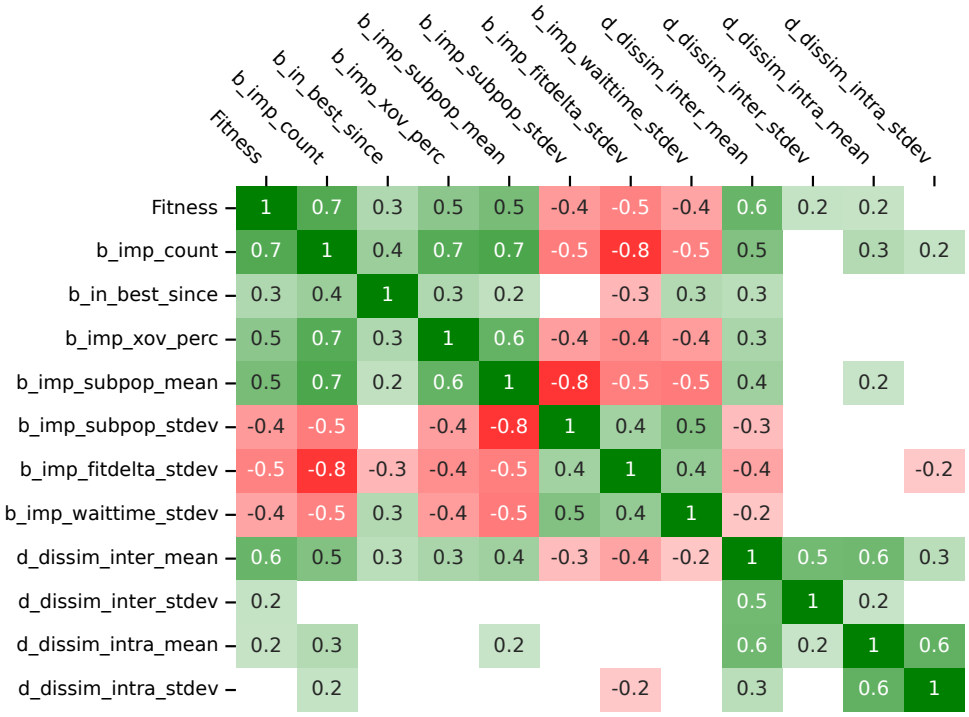
4 Effects of evolutionary dynamics on fitness

4.1 Correlations with fitness

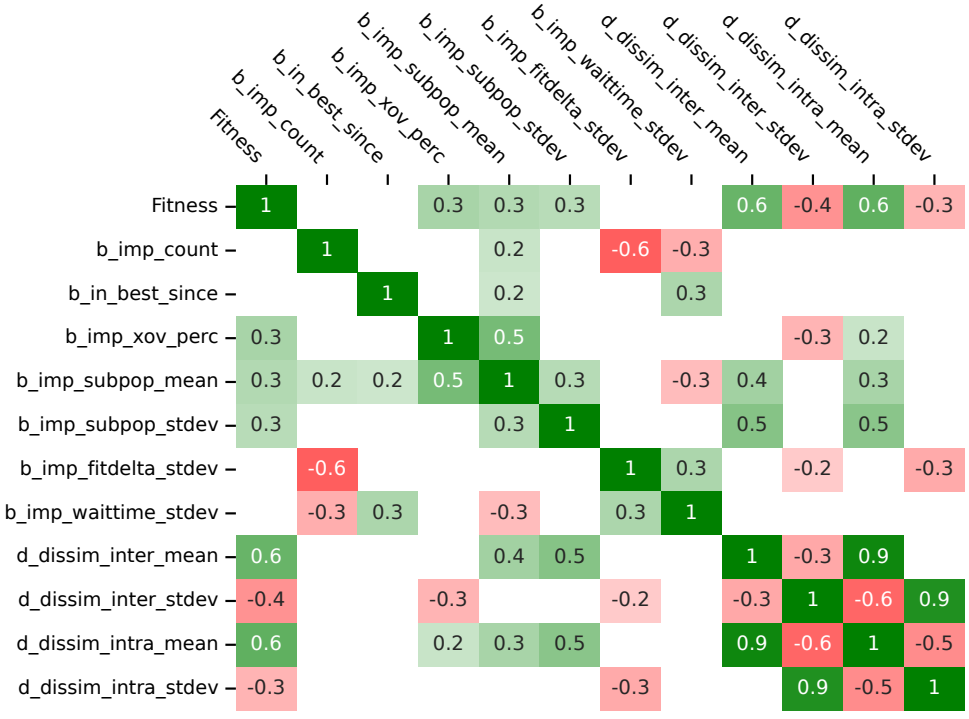
In order to identify the effect of the evolutionary dynamics on the effectiveness of the algorithm, we check the proposed features for Pearson's correlations – both with fitness and with one another. Correlations with $p > 0.05$ are omitted from the plots.

Fig. 3a shows the full correlation matrix for the **height_f1** task. The features that correlate with fitness the most are: *b_imp_count*, *d_dissim_inter_mean*, *b_imp_subpop_mean*, *b_imp_fitdelta_stdev* and *b_imp_xov_perc*. With an exception of *d_dissim_inter_mean*, all of these features are directly related to the improvements noted during the evolutionary run.

Fig. 3b presents the full correlation matrix for the **height_f9** task, which is noticeably different from the one for **height_f1**. Not only do most of the correlations appear to be statistically insignificant, but also some of them are the opposite of the ones observed in Fig. 3a. This is especially clear for the features based on dissimilarity – their correlations with fitness are stronger for **height_f9**. Moreover, while for **height_f1**, features based on both means and standard deviations of dissimilarity correlate positively with fitness, in the case of **height_f9** the standard deviations correlate negatively. This shows that the change of the genetic representation can noticeably affect the subpopulation dynamics.



(a) **height_f1** task.



(b) **height_f9** task.

Figure 3: Pearson correlation matrix for the **height_f1** (3a) and **height_f9** (3b) tasks. Correlations that are not statistically significant ($\alpha = 0.05$) are not shown.

4.2 Automated identification of evolutionary phenomena

Most of the features that strongly correlate with fitness also strongly correlate with one another, so their influence on fitness is not independent. Such correlations may indicate the existence of some more fundamental mechanisms that influence the success of evolutionary runs, with the proposed features being mere realizations (correlates) of these mechanisms. To uncover these mechanisms, should they exist, we use principal component analysis on the features of evolutionary runs (excluding fitness).

Fig. 4 presents the principal components for tasks **height_f1** and **height_f9**. The top row shows the percentage of the variance explained by each of the components, the middle matrix shows the coefficients of the features comprising the components, and the bottom row shows the correlation of each of the components with fitness. The components that were identified as significantly correlated with fitness can be interpreted in a meaningful way as *profiles of behavior* of the evolutionary process. As these profiles are guaranteed to be orthogonal to each other, they represent independent sources of fitness improvements during the evolution. They may be related to different successful paths through the fitness landscape of a problem, or leveraging different mechanisms present in the algorithm. Therefore, profiles of behavior can be used to reveal specific problems or positive effects that evolution encounters in the experiments, and may suggest possible improvements to the algorithm and its parametrization.

For both tasks, the first component (C_1) is very similar to the vector of correlations between features and fitness. This means that the variability of the inner dynamics is mainly driven by the mechanisms that have a direct effect on the quality of the search.

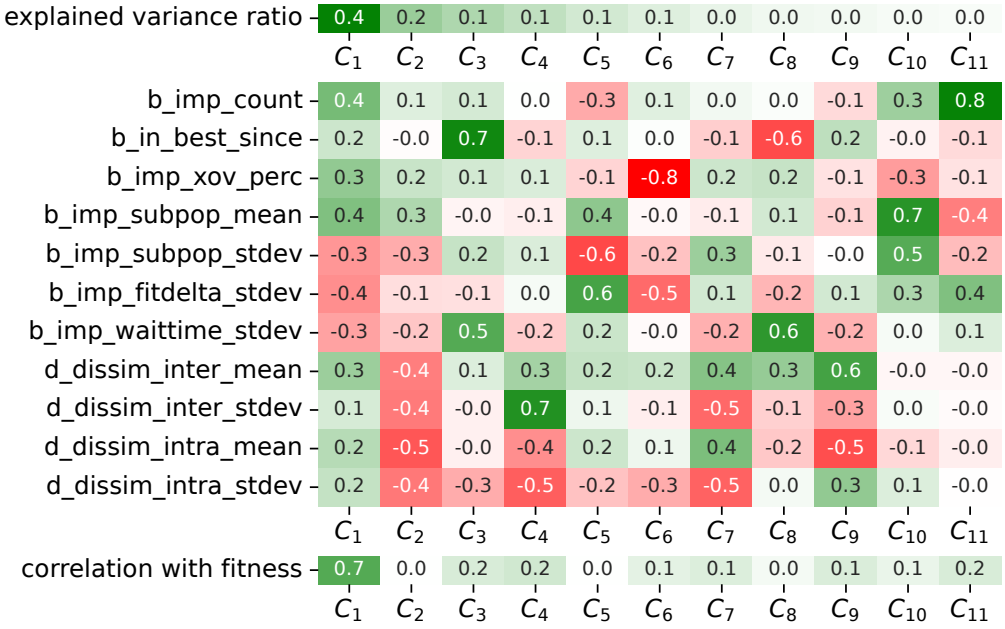
For the task **height_f1** (Fig. 4a), two components that correlate with fitness with a statistical significance of $\alpha = 0.1$ are C_1 ($p = 0.0$) and C_4 ($p = 0.073$). C_1 presents a profile of a smooth, mostly local optimization. The coefficients of the features imply evolution with many (*b_imp_count*) regular (*b_imp_waittime_stdev*) improvements of relatively constant sizes (*b_imp_fitdelta_stdev*), mostly from a few (*b_imp_subpop_stdev*) good subpopulations (*b_imp_subpop_mean*). C_4 presents a profile of an evolution where each subpopulation is focused on improving specific solutions – variability of the solutions within each subpopulation is consistently (*d_dissim_intra_stdev*) low (*d_dissim_intra_mean*), yet the diversity between subpopulations is high (*d_dissim_inter_mean*) and varies (*d_dissim_inter_stdev*). As such, this profile makes a good use of the subpopulation structure of convection selection. This suggests that in order to improve the performance of convection selection for **height_f1**, one should increase the selective pressure within the subpopulations, but also introduce additional mechanisms to increase the diversity between these subpopulations.

Fig. 4b demonstrates the principal components for **height_f9**. Two components that correlate with fitness with a statistical significance of $\alpha = 0.1$ are C_1 ($p = 0.0$) and C_6 ($p = 0.057$). C_1 presents a profile where the diversity of solutions is high, both within (*d_dissim_intra_mean*) and between (*d_dissim_inter_mean*) the subpopulations, yet its variance is relatively low (*d_dissim_inter_stdev*, *d_dissim_intra_stdev*). Improvements often originate as crossovers (*b_imp_xov_perc*) of solutions from a variety (*b_imp_subpop_stdev*) of good (*b_imp_subpop_mean*) subpopulations. This is a sign of an experiment which has not become trapped in a local optimum and is likely to still improve if it is given a chance. On the other hand, C_6 is a profile defined primarily by most improvements originating from the same fitness region (*b_imp_subpop_stdev*).

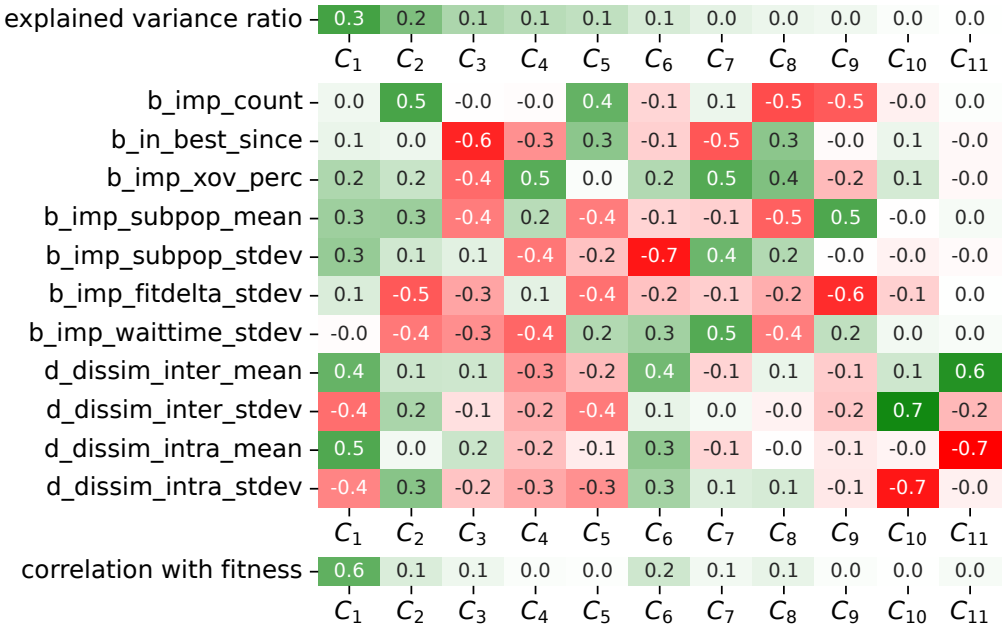
5 Summary and future work

In this paper we analyzed the relation between the inner evolutionary dynamics and the quality of the best solutions found during the optimization process in evolutionary algorithms with convection selection. In order to describe the subpopulation dynamics, we introduced a number of statistical features. The correlations between the features and the fitness of the best solutions were analyzed.

We used principal component analysis to find characteristic profiles of subpopulation dynamics, the presence of which can affect the performance of the algorithm. The interpretation of such profiles



(a) **height_f1** task.



(b) **height_f9** task.

Figure 4: Principal components from PCA on the values of the proposed features of evolutionary dynamics, in order of importance (from left to right), based on 100 independent evolutionary runs for **height_f1** (4a) and **height_f9** (4b) tasks.

may provide clues on how to improve the optimization algorithm or its components.

Our paper shows that interesting insights about the behavior of evolutionary algorithms can be gained from the analysis of statistical features of the population gathered during repeated evolutionary runs, even if these features are an effect of the stochasticity of the algorithm, and are beyond our direct control. Applying a similar methodology to other optimization tasks and other algorithms can serve as a useful tool aiding the adaptation of algorithms to specific classes of problems, and facilitating the explanation of interactions between optimization problems and algorithms.

Acknowledgements

This research was supported by the Polish Ministry of Education and Science, grant no. 0311/SBAD/0726.

References

[1] William H. E. Day. Properties of Levenshtein metrics on sequences. *Bulletin of Mathematical Biology*, 46(2):327–332, 1984.

[2] Maciej Komosinski and Konrad Miazga. Comparison of the tournament-based convection selection with the island model in evolutionary algorithms. *Journal of Comp. Sci.*, 32:106–114, 2018. doi:10.1016/j.jocs.2018.10.001.

[3] Maciej Komosinski and Szymon Ulatowski. Framsticks: Creating and understanding complexity of life. In M. Komosinski and A. Adamatzky, editors, *Artificial Life Models in Software*, chapter 5, pages 107–148. Springer, London, 2nd edition, 2009.

[4] 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.

[5] Maciej Komosinski and Szymon Ulatowski. Genetic encodings in framsticks, 2024. URL: http://www.framsticks.com/a/al_genotype.

[6] Hari Mohan Pandey, Ankit Chaudhary, and Deepti Mehrotra. A comparative review of approaches to prevent premature convergence in ga. *Applied Soft Computing*, 24:1047–1077, 2014.

[7] D. Sudholt. The benefits of population diversity in evolutionary algorithms: a survey of rigorous runtime analyses. *Theory of Evol. Comput.*, pages 359–404, 2020.

[8] Andrew James Turner and Julian Francis Miller. Neutral genetic drift: an investigation using cartesian genetic programming. *Genetic Programming and Evolvable Machines*, 16(4):531–558, 2015.