Working on the Structural Components of Evolutionary Approaches

Alexandros Tzanetos[1][a] and Jakub Kůdela[2][b]
[1]*Jönköping AI Lab (JAIL), Jönköping University, Gjuterigatan 5, 553 18 Jönköping, Sweden*
[2]*Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, Czech Republic*

Keywords: Evolutionary Computation, Swarm Intelligence, Nature-Inspired Intelligence, Mechanisms, Operators, Modular Algorithms.

Abstract: Several researchers have turned their attention to the structural components of Evolutionary Computation- and Swarm Intelligence-oriented approaches. This direction offers various opportunities, such as developing automatic design and configuration frameworks and integrating operators and mechanisms addressing known limitations. This work lists recent operators and discusses promising mechanisms found in existing nature-inspired approaches. It also discusses how these algorithmic components can be integrated into modular frameworks and how they can be assessed and benchmarked. The work aims to emphasize the importance of the research direction about nature-inspired mechanisms and operators in the Evolutionary Computation field.

1 INTRODUCTION

Evolutionary Computation (EC) and Swarm Intelligence (SI) offer a variety of approaches to deal with high-complexity real-world problems. Yet, not all algorithms from these fields constitute a novel search strategy. Most of them replicate the ideas introduced in the established ones, i.e., Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), and Genetic Algorithm (GA) (Tzanetos, 2023).

Moreover, several recently introduced algorithms contain a center-bias operator, making them unsuitable for optimization tasks (Kudela, 2022). Also, several other structural biases have been detected in such algorithms (Vermetten et al., 2022).

However, some algorithms contain promising mechanisms that can be used to overcome known limitations observed in stochastic nature-inspired algorithms (Thymianis and Tzanetos, 2022). More such mechanisms can potentially be found in the various nature-inspired algorithms and benefit other methods. The current work aims to provide some examples, discuss their benefits, and pinpoint the promising path ahead.

Furthermore, novel operators have been proposed to enhance the algorithms' performance. Our work briefly overviews such operators and discusses potential future directions.

Motivated by the recent works pinpointing the positive effect of improved operators and integrated mechanisms to nature-inspired approaches (NIAs), this position paper presents an initial step towards putting more emphasis on the research direction about nature-inspired mechanisms and operators in the EC field. Moreover, it concludes with a non-exhaustive list of future directions that we believe align with the topic's agenda.

The following section provides background on the heuristic nature of EC techniques. Section 3 lists recent operators and discusses promising mechanisms found in existing NIAs. Following the example of (Campelo and Aranha, 2021), we avoid citing the metaphor-based algorithms in which the mechanisms were found. Section 4 discusses the opportunity to put operators and mechanisms into a modular framework context and assess and benchmark those operators and mechanisms. Finally, in section 5, we mention some potential directions in which research can focus.

2 BACKGROUND

The Merriam-Webster dictionary defines the word "heuristic" as: "involving or serving as an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods", with

[a] https://orcid.org/0000-0002-1319-513X
[b] https://orcid.org/0000-0002-4372-2105

Tzanetos, A. and Kůdela, J.
Working on the Structural Components of Evolutionary Approaches.
DOI: 10.5220/0013083400003837
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Joint Conference on Computational Intelligence (IJCCI 2024)*, pages 375-382
ISBN: 978-989-758-721-4; ISSN: 2184-3236
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

its etymology being from the German “heuristisch”, New Latin “heuristicus”, and Greek “heuriskein” (εὑρίσκειν), meaning “to discover” (or “to happen upon by chance”).

Many of the EC techniques were conceived by this trial-and-error approach, with the useful mechanisms and operators for their applicability “discovered by chance”. For instance, the original description of Differential Evolution (DE) (Storn and Price, 1997) is given without any theoretical justification and analysis, treating it as a “pure heuristic”. Theoretical works on the convergence properties of the different operators in DE, such as the crossover ones in (Zaharie, 2009), came well after it became widely used. Also, the fact that DE has theoretically supported convergence problems for certain multimodal functions (Hu et al., 2016) does not bar it from being a successful and widely used method.

One of the oldest and still widely used stochastic heuristics is the Simulated Annealing (SA) algorithm (Kirkpatrick et al., 1983). Interestingly, for SA, the theoretically justified temperature control schemes (Hajek, 1988) do not work well in practice (Kochenderfer and Wheeler, 2019), and other “heuristically found” schemes are used instead.

In 1965, the Nelder-Mead (NM) simplex method, one of the oldest deterministic optimization heuristics, was introduced (Nelder and Mead, 1965). Although the mechanisms of the NM method are not very complex, it took more than 30 years from its inception before the first notable theoretical work about its convergence properties was published (Lagarias et al., 1998). In the paper, the authors prove that the method converges for strictly convex functions in 1D but not 2D (by counterexample). The paper ends with the following statement: “*Our general conclusion about the Nelder-Mead algorithm is that the main mystery to be solved is not whether it ultimately converges to a minimizer—for general (nonconvex) functions, it does not—but rather why it tends to work so well in practice by producing a rapid initial decrease in function values.*” Further research led to convergence proofs for 2D, but only for restricted versions on strictly convex functions (Lagarias et al., 2012), with the technique of the proof based on treating the method as a discrete dynamical system. And new results (such as convergence proofs for dimensions greater than 8) are expected to need techniques that are not yet developed (Galántai, 2022).

What these results show is that, in many cases, the theoretical justifications for EC methods (and heuristics in general) might not lead to useful variants of such methods (such as in the SA case) or may not be derivable by currently available techniques (such as in the NM case). It is not hard to imagine a simple mechanism, with inner workings akin to the $3n+1$ problem, that will elude theoretical analysis (Guy, 2004).

Nevertheless, there has been undeniable progress in the theoretical works, mainly focusing on discrete settings with “simpler” EC methods for which tools such as drift analyses can be used (Doerr and Neumann, 2021). For the continuous setting and the “more involved” techniques, experimental methods for runtime analysis are available (Huang et al., 2019).

3 NATURE-INSPIRED MECHANISMS AND OPERATORS

The difference between a *mechanism* and an *operator* may not be obvious. Both terms appear in the literature, usually to describe algorithmic components.

A suitable definition for the word **mechanism**, given by (Ross et al., 2022), states that it is “*a process or system that is used to produce a particular result*”.

No standard definition exists for the word **operator** within the sciences concept. In mathematics, an operator refers to a mapping or function that acts on elements of a space to produce elements of another space. In computer programming, operators are constructs defined within programming languages that behave like functions. A common ground in the above definitions is the concept of a function. However, we cannot claim that an operator is a function. The Merriam-Webster dictionary definition seems more suitable: “*a single step performed by a computer in the execution of a program*”.

A more open-ended definition for Evolutionary Operators is “*a total parameter-alteration operation on the parameterization of generalized coordinates of a system along its dynamic path*” given by (Öz, 2005), where the theoretical aspects of variational and extremum principle derived from various sciences are studied.

To distinguish the two concepts within the EC field, we can say that a **mechanism** is a process applied to the whole population of candidate solutions. In contrast, an **operator** is a single-step operation applied to some or selected candidate solutions.

3.1 Traditional Operators

The original idea of operators comes from the evolutionary operations included in the Darwinian-like evolutionary process in the first evolutionary approaches (De Jong, 2012). These operations correspond to (a) asexual reproduction, i.e., the *mutation* operator, (b) sexual reproduction, i.e., the *crossover* operator, and (c) natural selection, i.e., the *selection* operator.

The *mutation* and *crossover* operators, also called reproductive operators, create a mix of local and global search. The *selection* operator aims to balance exploration and exploitation effectively.

The above evolutionary operators are very simple operations. Indeed, they involve combining elements of more than one parent solution, altering some of the current solution's elements, and selecting which candidate solutions will be passed to the next generation. The rationale behind these operators is to tackle pseudo-Boolean optimization problems:

$$f : \{0,1\}^d \to \mathbb{R} \quad (1)$$

which consisted of d binary decision variables. The binary value solution space makes the above evolutionary operators proper for searching different combinations of a given solution.

3.2 Recently Presented Operators

Several researchers have developed operators that overcome the limitations of the original ones. Such limitations are infeasible solutions created by the reproduction operators, low-quality solutions transferred to the new population due to a poor selection process, and premature convergence due to low diversity in the population. Below, we present an example for each evolutionary operator, i.e., crossover, mutation, selection, and one example of a population management operator.

(Sharma et al., 2021) introduced the idea of *ensembled crossover* for evolutionary algorithms, considering that different crossovers are suitable for different optimization problems. As a result, the authors mentioned that the *ensembled crossover* is more robust on various optimization problems with different characteristics such as variable dependency or multimodality. The difference from the original crossover operator is that the crossover probability is drawn from various distributions, e.g., uniform, epsilon, exponential, etc. Each distribution has the same probability of being selected at the beginning. The algorithm updates the probabilities based on the proportion of survivors to the created offspring members.

The *Neighbor-Hop mutation* is a mutation operator where a gene is replaced randomly by one of its neighbor nodes in a graph (Chawla and Cheney, 2023). The authors applied their *Neighbor-Hop mutation*-based GA to the Influence Maximization Problem, i.e., finding the set of k most influential nodes in a social network. The solution representation of the GA for that problem is a sequence of k nodes from the given graph. The introduced mutation operator is suitable for the application since it explores different seed node permutations without causing any infeasible solutions. That would not be the case if the problem was the Travelling Salesman Problem. The *Neighbor-Hop mutation* is an example of an operator introduced for a specific problem type.

The *fitness-distance balance* is a recent selection operator introduced by (Kahraman et al., 2020) that enables the proper determination of candidates with the highest potential to improve the search process. First, the distance between the candidate and the best solution is calculated and stored in a distance vector. Then, the distance vector and the fitness vector are normalized. The fitness vector contains the fitness values of the candidate solutions in the same order as the distance vector. The Fitness-Distance Balance (FDB) score is calculated as:

$$S_{FDB} = w \cdot norm_{F_i} + (1-w) \cdot norm_{D_i} \quad (2)$$

where $norm_{D_i}$ is the normalized distance value of the i-th candidate solution and $norm_{F_i}$ is the corresponding candidate solution's normalized fitness value. The weight coefficient $w \in (0,1)$ determines the impact of fitness and distance values.

An alternative way to calculate the FDB score is:

$$S_{FDB} = norm_{F_i} \cdot norm_{D_i} \quad (3)$$

Another category of operators consists of the ones considering population diversity. They monitor the population diversity and perform changes, e.g., in the algorithm's selection operator (Strzeżek et al., 2015), or in the population size.

(Morales-Castaneda et al., 2022) introduced a new set of evolutionary operators that falls within the second case. These operators permit the management of the population size to avoid search stagnation and allow more efficient exploitation. To prevent search stagnation, the *Stagnation Correction Operator* introduces new candidate solutions using a dimension-wise diversity measurement metric (Hussain et al., 2019). To improve the exploitation efficiency, the *Exploitation Efficiency Operator* compares the current diversity with the maximum diversity previously found, and based on a predefined diversity percentage, the population is reduced. In their work, the authors use 10% as the diversity percentage that indicates the algorithm is entering its exploitation phase, and they reduce the population by $1/5$. For a diversity percentage equal to 2%, indicating that the exploitation phase is deep in progress, the population is more significantly reduced by $3/5$.

Diversity-based operators have shown promising results (Cheng et al., 2021; Wang et al., 2023). Work on such operators can be extended by considering other diversity measures, such as the attraction basins-based measure introduced by (Jerebic et al., 2021).

3.3 Promising Mechanisms

3.3.1 Cyclic Universe Theory

The cyclic universe theory (Steinhardt and Turok, 2002) inspired the Big Bang - Big Crunch (BB-BC) algorithm. According to this theory, the universe undergoes endless cycles of expansion and cooling, each beginning with a "Big Bang" and ending in a "Big Crunch". The Big Bang phase is the expansion phase, whereas, in the Big Crunch phase, the gravitational attraction of matter causes the universe to collapse back in. The two phases described above comprise the BB-BC algorithm.

During the Big Bang phase, the candidate solutions are randomly placed in the solution space using the following formula:

$$x^d_{new} = x^d_c + \frac{l^d \times rand}{k} \tag{4}$$

where l^d is the upper limit for the d-th dimension, *rand* is a random number uniformly distributed and k is the current iteration of the algorithm. This equation considers the "center of mass" $\overrightarrow{x_c} = [x^1_c, x^2_c, ..., x^{ND}_c]$, where *ND* is the number of the problem's dimensions.

In the Big Crunch phase, the candidate solutions converge to a single point, i.e., the center of mass, calculated as:

$$x^d_c = \frac{\sum_{i=1}^{N} {}^1\!/_{f^i}\, x^d_i}{\sum_{i=1}^{N} {}^1\!/_{f_i}} \tag{5}$$

where x^d_i denotes the position of the i-th candidate solution in the d-th dimension, f_i denotes the value of the i-th candidate solution in the fitness function and N is the population size.

The BB-BC algorithm iterates between these two phases. The population of the previous generation is used to calculate the center of mass using eq. 5 and then, new candidate solutions are generated around this center of mass using eq. 4.

This cyclic universe process is the equivalent of killing the population in each generation and generating a new one. The advantage of this concept is that the center of mass contains information from all candidate solutions. In EC, and especially in evolutionary algorithms, the new population is randomly created. On the other hand, the continuous iteration between "killing the population" and "creating a new one" has the drawback of not letting any candidate solutions evolve.

3.3.2 Mine Explosion Dynamics

The Mine Blast Algorithm (MBA) contains a mechanism inspired by the mine explosion dynamics. Specifically, considering that $\overrightarrow{X} = \{X_1, X_2, ..., X_m\}$ represents the location of a mine and that N_s shrapnel pieces are produced by the mine bomb explosion causing another mine to explode at location $X_{e(n+1)}$, calculated as:

$$X_{e(n+1)} = d_{n+1} \times \cos\theta \tag{6}$$

where θ is the angle of the shrapnel pieces, which is a constant value and is calculated using $\theta = 360/N_s$ and d_{n+1} is the distance of the thrown shrapnel pieces, calculated as:

$$d_{n+1} = d_n \cdot |randn|^2 \tag{7}$$

where *randn* is a normally distributed random number, and n is a number defined as $n \in 1, 2, ..., N_s$.

The above formulation describes the algorithm's exploration phase. Additionally, another operator is set for the algorithm's exploitation phase. In that case, the shrapnel's position is calculated as:

$$X_{n+1} = X_{e(n+1)} + \exp\left(-\sqrt{\frac{m_{n+1}}{d_{n+1}}}\right) \cdot X_n \tag{8}$$

where d_{n+1} and m_{n+1} denote the distance and the direction (slope) of the thrown shrapnel pieces in each iteration, respectively. Also, $X_{e(n+1)}$ denotes the location of exploding mine bomb collided by shrapnel, given by:

$$X_{e(n+1)} = d_n \cdot rand \cdot \cos\theta \tag{9}$$

where *rand* is a uniformly distributed random number and θ is the angle of the shrapnel pieces which is a constant value and is calculated using $\theta = 360/N_s$, where N_s denotes the number of shrapnel pieces.

The distance d_{n+1} of the thrown shrapnel pieces is calculated as:

$$d_{n+1} = \sqrt{(X_{n+1} - X_n)^2 + (F_{n+1} - F_n)^2} \tag{10}$$

where F denotes the quality of each particle.

The direction m_{n+1} of the thrown shrapnel pieces is given by:

$$m_{n+1} = \frac{F_{n+1} - F_n}{X_{n+1} - X_n} \tag{11}$$

3.3.3 Negatively Charged Stepped Leader

The Lightning Search Algorithm (LSA)'s central concept is inspired by the negatively charged stepped leader moving from a cloud to the earth (Dul'zon et al., 1999), i.e., the phenomenon of separations of the luminous channel (i.e., projectile) that propagates through ionized paths, creating this tree structure of lightning.

Three projectile types are considered in LSA, i.e., transition, space, and lead. The transition projectiles are used to create the first-step leader population N, the space projectiles are considered to reach the best leader position, and the lead projectiles represent the best position among the N step leaders.

Transition projectiles create a random population of N first-step leader uniformly distributed. They are also used to create new projectiles when necessary.

The space projectile p_i^S is calculated as:

$$p_{i,new}^S = p_i^S \pm exprand(\mu_i) \qquad (12)$$

where $exprand(\mu_i)$ denotes an exponentially distributed random number. The operator $\pm$ in eq. 12 denotes that $exprand(\mu_i)$ should be subtracted if p_i is negative, but it should be added if p_i is positive. This operation is used to avoid any change of direction in the solution space since $exprand(\mu_i)$ returns only positive values.

This step separates the projectile into stepped leaders. If $p_{i,new}^S$ provides a good solution, the corresponding stepped leader will be further separated. Otherwise, it will remain unchanged. If $p_{i,new}^S$ extends all stepped leaders beyond the recent most extended leader, it becomes the lead projectile.

Finally, the lead projectile p_i^L is calculated as:

$$p_{i,new}^L = p_i^L \pm normrand(\mu_L, \sigma_L) \qquad (13)$$

where $normrand(\mu_L, \sigma_L)$ denotes a normally distributed random number. This step is performed for exploitation. As for the space projectile, if $p_{i,new}^L$ provides a better solution, it will be extended until no better solution is found.

3.3.4 Spiral Dynamics

The Spiral Dynamic Algorithm (SDA) introduced a concept inspired by spiral phenomena observed in nature, such as galaxies, hurricanes, and tornadoes. The concept is a spiral model that provides dynamic step size for each candidate solution when searching the solution space. The step size is larger at the beginning to enable exploration and gets smaller during the algorithm to focus on exploitation. The step size is defined based on a spiral radius r and the spiral rotation angle θ, which are open parameters.

A similar concept was introduced in Hurricane-based Optimization Algorithm (HOA) where solutions are divided into $j = d - 1$ groups based on the problem's dimensions d. The candidate solution performs a rotation in each dimension defined by the group $k = i \mod (d-1)$ it belongs to. In this model, more parameters are used but based on the same concept, i.e., radial and angular coordinates.

The spiral movement concept also inspired the Circular Water Wave (CWW) algorithm. The step size is calculated using a simpler equation compared to SDA and HOA. The spiral model used in this algorithm considers a radius r_i, a random value w that is used as a coefficient, and the wave number c_j, where $1 \leq j \leq m$, and m denotes the maximum number of wave circles, i.e., iterations of the algorithm.

Even though different models are introduced in the three algorithms mentioned above, the mechanism is the same. That is, a radius and a rotation angle define the step sizes, which are performed towards an "eye", i.e., a central point.

3.4 Discussion

As pointed out by (Tzanetos, 2023), only the *cyclic universe mechanism* seems consistent with the phenomenon inspired from. Also, the *spiral dynamics mechanism* utilizes the mathematical background of spiral dynamics. The rest of the mechanisms mentioned above are simplifications of a phenomenon.

Nevertheless, the mechanisms mentioned in the current work can be useful components for modified or improved versions of established approaches. (Thymianis and Tzanetos, 2022) showed that the integration of the *cyclic universe mechanism* and the *mine explosion mechanism* improve the exploration ability of fast converging approaches, such as PSO.

We believe that a similar value can be found in the rest of the above mechanisms. The *Negatively Charged Stepped Leader* seems a promising exploration strategy (Tzanetos, 2023), too. Moreover, it could be combined with a population management operator that eliminates non-promising solutions before further separation into stepped leaders occurs. The *spiral dynamics mechanism* seems a promising exploitation strategy, i.e., a local search component to intensify the search around local optima.

Furthermore, other promising mechanisms can be found in the literature. A crucial question is how we can identify such mechanisms. This is not a trivial task since using metaphor-based language does not enable researchers to fully comprehend a method's mathematical background (Aranha et al., 2022).

While mechanisms constitute different stepwise processes, most of the existing operators are variations of the evolutionary operators, i.e., crossover, mutation, selection, or population management techniques. Research on new operators is more established than that on mechanisms. Apart from the numerous papers found in the literature, relevant workshops exist.

For example, dedicated workshops enable researchers to investigate the *selection* aspect in EC. The most recent such workshops are the *Workshop on Selection* [1] and the *Workshop on Search and Selection in Continuous Domains* [2].

4 MODULAR FRAMEWORKS

Identifying the individual mechanisms and operators of EC methods enables the creation and investigation of modular frameworks, which facilitate the (either manual or automatic) design of high-performance implementations of EC methods (Camacho-Villalón et al., 2023). Among the first modular frameworks was ParadisEO (Cahon et al., 2004), which contains four main modules/objects for the composition of EC methods: estimation of distribution objects (for estimation of distribution methods), evolving objects (for population-based methods), moving objects (for local search methods), and multiobjective evolving objects (for multiobjective methods).

Recently, there has been an increased interest in creating modular frameworks around established EC methods, with a focus on the inclusion of the most widely used/representative mechanisms and operators for that method. The most notable ones are the modular frameworks for ACO (López-Ibáñez et al., 2018), PSO (Camacho-Villalón et al., 2021), Covariance matrix adaptation evolution strategy (CMAES) (de Nobel et al., 2021), and DE (Vermetten et al., 2023).

There is a current debate about the different merits of manual and automatic design of EC methods using modular frameworks (Camacho-Villalón et al., 2023). The manual approaches rely heavily on an "expert designer" who understands not only the inner workings of the algorithms but also the problem to which they are applied. On the other hand, the automatic design of EC methods usually utilizes configuration tools such as SMAC (Hutter et al., 2011) or irace (López-Ibáñez et al., 2016). However, these tools still require some input from the user, such as selecting a set of relevant operators and mechanisms. For white-box problems, where the mathematical description of the problem to be solved is known, the possibility of using a pre-processing tool that scans the problem description and returns a reasonable set of possible operators for the automatic selection tool (hence requiring even less expertise from the user) is still a not very well explored area. Recent results of using the various automatic configuration tools in discrete (Aziz-Alaoui et al., 2021; Ye et al., 2022) are very promising.

Another relatively under-explored line of research lies in assessing "usefulness" and benchmarking the individual modules and their combinations. In (Nikolikj et al., 2024), this assessment was performed with the utilization of the functional ANOVA (f-ANOVA) framework (Hutter et al., 2014). Although the experiments were relatively large-scale in terms of compared variants (324 CMAES and 576 DE variants), they were only small-scale in terms of dimensions (only dimensions 5 and 30 were investigated) and computational budgets ($100 \times ND$ - $1500 \times ND$ function calls). Another approach utilizing the Search Trajectory Networks, population diversity, and anytime hypervolume values for the assessment of the impact of the modules in the multiobjective setting was described in (Lavinas et al., 2024). To assess budget-dependent mechanisms/operators (such as linear population reduction schemes), the standard anytime performance measures should be used cautiously (Tušar et al., 2017).

5 CONCLUSIONS AND FUTURE DIRECTIONS

This position paper presents an initial step towards putting more emphasis on the research direction about nature-inspired mechanisms and operators in the EC field. We believe this direction offers various opportunities, such as extending the automatic design and configuration frameworks by including proper mechanisms and operators, integrating existing operators and mechanisms into established algorithms to address their limitations, developing new proper operators, identifying other promising mechanisms, and investigating performance measures to assess the effect of these algorithmic components.

Recent works pinpointed the positive effect of improved operators and integrated mechanisms to NIAs. Along those lines, we listed four recently developed operators and four promising mechanisms found in existing NIAs. We discussed the benefits of those selected operators and mechanisms. We also discussed how these algorithmic components can be integrated into modular frameworks and how they can be assessed and benchmarked.

As mentioned above, we see this work as a starting point in the research agenda on nature-inspired mechanisms and operators. Therefore, we list a few future directions below that we believe align with the topic's

[1] https://www.yorku.ca/sychen/research/workshops/CEC2021_Selection_Workshop.html

[2] https://www.yorku.ca/sychen/research/workshops/CEC2024_Search_and_Selection_Workshop.html

agenda. We do not claim that these directions are exhaustive. We invite interested researchers to extend this agenda.

Following the recent progress in the theoretical background of evolutionary approaches, the mathematical and theoretical analysis of mechanisms and operators included in EC and SI algorithms could provide more insight into these algorithmic components and their effects.

Identifying and implementing other mechanisms being part of existing EC and SI algorithms will extend the options of algorithmic components. Identifying several mechanisms with various characteristics enables selecting the most suitable to address an algorithm's known limitation. Also, EC researchers could focus on developing improved operators that overcome the defects of the existing ones.

Furthermore, a smooth integration between different automatic operator selection tools with the existing modular frameworks could help identify the usefulness of both the individual mechanisms and operators and their interplay. The methodologies for properly assessing and benchmarking these components are currently being developed and tested, leaving ample room for extensions and improvements.

Last but not least, applying the modified algorithms to real-world problems is equally important to the theoretical work. After all, the ultimate goal of developing an algorithm is to solve a problem. Thus, investigating the modified algorithms' performance on real-world problems will reveal which components may be more suitable for specific problem types.

ACKNOWLEDGEMENTS

The research work of J. Kůdela was supported by a grant (No. FSI-S-23-8394) from the *IGA Brno University of Technology* and by the project (no. 24-12474S) funded by the *Czech Science Foundation*.

REFERENCES

Aranha, C., Camacho Villalón, C. L., Campelo, F., Dorigo, M., Ruiz, R., Sevaux, M., Sörensen, K., and Stützle, T. (2022). Metaphor-based metaheuristics, a call for action: the elephant in the room. *Swarm Intelligence*, 16(1):1–6.

Aziz-Alaoui, A., Doerr, C., and Dreo, J. (2021). Towards large scale automated algorithm design by integrating modular benchmarking frameworks. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1365–1374.

Cahon, S., Melab, N., and Talbi, E.-G. (2004). Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of heuristics*, 10(3):357–380.

Camacho-Villalón, C. L., Dorigo, M., and Stützle, T. (2021). Pso-x: A component-based framework for the automatic design of particle swarm optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 26(3):402–416.

Camacho-Villalón, C. L., Stützle, T., and Dorigo, M. (2023). Designing new metaheuristics: manual versus automatic approaches. *Intelligent Computing*, 2:0048.

Campelo, F. and Aranha, C. (2021). Sharks, zombies and volleyball: Lessons from the evolutionary computation bestiary. In *LIFELIKE Computing Systems Workshop 2021*. CEUR-WS. org.

Chawla, A. and Cheney, N. (2023). Neighbor-hop mutation for genetic algorithm in influence maximization. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, pages 187–190.

Cheng, J., Pan, Z., Liang, H., Gao, Z., and Gao, J. (2021). Differential evolution algorithm with fitness and diversity ranking-based mutation operator. *Swarm and Evolutionary Computation*, 61:100816.

De Jong, K. (2012). Generalized evolutionary algorithms. In Rozenberg, G., Bäck, T., and Kok, J. N., editors, *Handbook of Natural Computing*, pages 625–635. Springer Berlin Heidelberg, Berlin, Heidelberg.

de Nobel, J., Vermetten, D., Wang, H., Doerr, C., and Bäck, T. (2021). Tuning as a means of assessing the benefits of new ideas in interplay with existing algorithmic modules. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1375–1384.

Doerr, B. and Neumann, F. (2021). A survey on recent progress in the theory of evolutionary algorithms for discrete optimization. *ACM Transactions on Evolutionary Learning and Optimization*, 1(4):1–43.

Dul'zon, A., Lopatin, V., Noskov, M., and Pleshkov, O. (1999). Modeling the development of the stepped leader of a lightning discharge. *Technical physics*, 44(4):394–398.

Galántai, A. (2022). Convergence of the nelder-mead method. *Numerical Algorithms*, pages 1–30.

Guy, R. (2004). *Unsolved problems in number theory*, volume 1. Springer Science & Business Media.

Hajek, B. (1988). Cooling schedules for optimal annealing. *Mathematics of operations research*, 13(2):311–329.

Hu, Z., Su, Q., Yang, X., and Xiong, Z. (2016). Not guaranteeing convergence of differential evolution on a class of multimodal functions. *Applied Soft Computing*, 41:479–487.

Huang, H., Su, J., Zhang, Y., and Hao, Z. (2019). An experimental method to estimate running time of evolutionary algorithms for continuous optimization. *IEEE Transactions on Evolutionary Computation*, 24(2):275–289.

Hussain, K., Salleh, M. N. M., Cheng, S., and Shi, Y. (2019). On the exploration and exploitation in popular swarm-based metaheuristic algorithms. *Neural Computing and Applications*, 31(11):7665–7683.

Hutter, F., Hoos, H., and Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. In *International conference on machine learning*, pages 754–762. PMLR.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*, pages 507–523. Springer.

Jerebic, J., Mernik, M., Liu, S.-H., Ravber, M., Baketarić, M., Mernik, L., and Črepinšek, M. (2021). A novel direct measure of exploration and exploitation based on attraction basins. *Expert Systems with Applications*, 167:114353.

Kahraman, H. T., Aras, S., and Gedikli, E. (2020). Fitness-distance balance (fdb): a new selection method for meta-heuristic search algorithms. *Knowledge-Based Systems*, 190:105169.

Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

Kochenderfer, M. J. and Wheeler, T. A. (2019). *Algorithms for optimization*. Mit Press.

Kudela, J. (2022). A critical problem in benchmarking and analysis of evolutionary computation methods. *Nature Machine Intelligence*, 4(12):1238–1245.

Lagarias, J. C., Poonen, B., and Wright, M. H. (2012). Convergence of the restricted nelder–mead algorithm in two dimensions. *SIAM Journal on Optimization*, 22(2):501–532.

Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E. (1998). Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM Journal on optimization*, 9(1):112–147.

Lavinas, Y., Ladeira, M., Ochoa, G., and Aranha, C. (2024). Multiobjective evolutionary component effect on algorithm behaviour. *ACM Transactions on Evolutionary Learning and Optimization*, 4(2):1–24.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., and Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

López-Ibáñez, M., Stützle, T., and Dorigo, M. (2018). Ant colony optimization: A component-wise overview. In Martí, R., Pardalos, P. M., and Resende, M. G. C., editors, *Handbook of heuristics*, volume 932, pages 371–407. Springer International Publishing, Cham.

Morales-Castaneda, B., Maciel-Castillo, O., Navarro, M. A., Aranguren, I., Valdivia, A., Ramos-Michel, A., Oliva, D., and Hinojosa, S. (2022). Handling stagnation through diversity analysis: A new set of operators for evolutionary algorithms. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–7. IEEE.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4):308–313.

Nikolikj, A., Kostovska, A., Vermetten, D., Doerr, C., and Eftimov, T. (2024). Quantifying individual and joint module impact in modular optimization frameworks. *arXiv preprint arXiv:2405.11964*.

Öz, H. (2005). Evolutionary energy method (eem): An aerothermoservoelectroelastic application. In *Variational and Extremum Principles in Macroscopic Systems*, pages 641–671. Elsevier.

Ross, R., Winstead, M., and McEvilley, M. (2022). Engineering Trustworthy Secure Systems. Technical Report NIST Special Publication (SP) 800-160 Vol. 1 Rev. 1, National Institute of Standards and Technology.

Sharma, S., Blank, J., Deb, K., and Panigrahi, B. K. (2021). Ensembled crossover based evolutionary algorithm for single and multi-objective optimization. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1439–1446. IEEE.

Steinhardt, P. J. and Turok, N. (2002). A cyclic model of the universe. *Science*, 296(5572):1436–1439.

Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11:341–359.

Strzeżek, A., Trammer, L., and Sydow, M. (2015). Divergene: Experiments on controlling population diversity in genetic algorithm with a dispersion operator. In *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 155–162. IEEE.

Thymianis, M. and Tzanetos, A. (2022). Is integration of mechanisms a way to enhance a nature-inspired algorithm? *Natural Computing*, pages 1–21.

Tušar, T., Hansen, N., and Brockhoff, D. (2017). Anytime benchmarking of budget-dependent algorithms with the coco platform. In *IS 2017-International multiconference Information Society*, pages 1–4.

Tzanetos, A. (2023). Does the field of nature-inspired computing contribute to achieving lifelike features? *Artificial Life*, 29(4):487–511.

Vermetten, D., Caraffini, F., Kononova, A. V., and Bäck, T. (2023). Modular differential evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 864–872.

Vermetten, D., van Stein, B., Caraffini, F., Minku, L. L., and Kononova, A. V. (2022). Bias: a toolbox for benchmarking structural bias in the continuous domain. *IEEE Transactions on Evolutionary Computation*, 26(6):1380–1393.

Wang, P., Xue, B., Liang, J., and Zhang, M. (2023). Feature selection using diversity-based multi-objective binary differential evolution. *Information Sciences*, 626:586–606.

Ye, F., Doerr, C., Wang, H., and Bäck, T. (2022). Automated configuration of genetic algorithms by tuning for anytime performance. *IEEE Transactions on Evolutionary Computation*, 26(6):1526–1538.

Zaharie, D. (2009). Influence of crossover on the behavior of differential evolution algorithms. *Applied soft computing*, 9(3):1126–1138.