Düzce University Journal of Science & Technology

Research Article

Improved Slime-Mould-Algorithm with Fitness Distance Balance-based Guiding Mechanism for Global Optimization Problems[1]

Çağrı Suiçmez [a,*], Hamdi Tolga Kahraman [b], Cemal Yılmaz[c], Mehmet Fatih Işık[d], Enes Cengiz[e]

[a] *Electrical Electronics Engineering, Technology Faculty, Gazi University, Ankara, TURKEY*
[b] *Software Engineering, Of Technology Faculty, Karadeniz Technical University, Trabzon, 61080, TURKEY*
[c] *Mingachevir State University, Mingachevir, AZERBAIJAN*
[d] *Electrical Electronics Engineering, Faculty of Engineering, Hitit University, Çorum, TURKEY*
[e] *Mechatronics Engineering, Technology Faculty, Afyon Kocatepe University, Afyonkarahisar, TURKEY*
* *Corresponding author's e-mail address: cagrisuicmez@gazi.edu.tr*

DOI: 10.29130/dubited.1016209

ABSTRACT

In this study, the performance of Slime-Mould-Algorithm (SMA), a current Meta-Heuristic Search algorithm, is improved. In order to model the search process lifecycle process more effectively in the SMA algorithm, the solution candidates guiding the search process were determined using the fitness-distance balance (FDB) method. Although the performance of the SMA algorithm is accepted, it is seen that the performance of the FDB-SMA algorithm developed thanks to the applied FDB method is much better. CEC 2020, which has current benchmark problems, was used to test the performance of the developed FDB-SMA algorithm. 10 different unconstrained comparison problems taken from CEC 2020 are designed by arranging them in 30-50-100 dimensions. Experimental studies were carried out using the designed comparison problems and analyzed with Friedman and Wilcoxon statistical test methods. According to the results of the analysis, it has been seen that the FDB-SMA variations outperform the basic algorithm (SMA) in all experimental studies.

Keywords: *Meta-heuristic search; slime mould algorithm; fitness-distance balance (FDB); benchmark problems.*

Global Optimizasyonu için Uygunluk Mesafe Dengesi Tabanlı Rehber Mekanızmasıyla Slime Mould Optimize Edicinin İyileştirilmesi

ÖZ

Bu çalışmada güncel bir Meta-Heuristic Search algoritması olan Balçık-Küfü Algoritması (SMA) performansının iyileştirmesi yapılmaktadır. SMA algoritmasında arama süreci yaşam döngüsü sürecini daha etkili bir şekilde modelleyebilmek için arama sürecine rehberlik eden çözüm adayları uzaklık-uygunluk dengesi (fitness-distance balance, FDB) yöntemi kullanılarak belirlenmiştir. Her ne kadar SMA algoritmasının performansı kabul görse de uygulanan FDB yöntemi sayesinde geliştirilen FDB-SMA algoritmasının performansının çok daha iyi olduğu görülmektedir. Geliştirilen FDB-SMA algoritmasının performansını test etmek için güncel benchmark sorunları olan CEC 2020 kullanılmıştır. CEC 2020'den alınan 10 farklı kısıtsız kıyaslama problemi 30-50-100 boyutlarında düzenlenerek tasarlanmıştır. Deneysel çalışmalar tasarlanan kıyaslama problemleri kullanılarak gerçekleştirilmiş ve Friedman ve Wilcoxon istatistiksel test yöntemleri ile analiz edilmiştir. Analiz sonuçlarına göre FDB-SMA varyasyonlarının tüm deneysel çalışmalarda temel algoritmaya (SMA) göre daha üstün bir performans gösterdiği görülmüştür.

Anahtar Kelimeler: *Meta-sezgisel arama; Slime Mould algoritması; uygunluk-mesafe dengesi (FDB); kıyaslama problemleri.*

[1]ICAIAME 2021 konferansında özet metin olarak sunulmuştur.

Received: 29/10/2021, Revised: 18/11/2021, Accepted: 21/11/2021

I. INTRODUCTION

Stochastic optimization algorithms are divided into Heuristic and Meta-Heuristic Search (MHS). Heuristic algorithms depend more on the type of problem. MHS-based algorithms, on the other hand, are used to obtain spherical or near-spherical optimum solutions as an alternative to mathematical approaches that are independent of the problem type [1,2]. In addition, Meta-Heuristic algorithms have become more advantageous in terms of better performance and computational cost than deterministic algorithms in many optimization problems in recent years [3,4,5].

Engineers and philosophers have always been inspired by nature, and many MHS-based algorithms have used this inspiration [6]. Evolutionary Algorithm (EA) developed by Fogel et al. [7]. Genetic Algorithm [8], developed in the 1960s and 1970s, inspired by the field of biology. Taboo Search (TS) inspired by animal behavior [9]. Particle swarm optimization inspired by bird flocks [10,11]. In addition, another developed MHS algorithm, Simulated Annealing (SA) algorithm, was inspired by the annealing process of metals [12,13].

Big Bang–Big Crunch algorithm (BB–BC) proposed by Erol and Eksin and developed by Kaveh and Talatahari [14,15] and Gravity Search Algorithm (GSA) developed by Rashedi et al. [16]. These algorithms are developed using the laws of physics. Along with these, there are algorithms inspired by music. For example, the Harmony Search (HS) algorithm [17]. Artificial Bee Colony (ABC) developed by being influenced by the food finding process of bees [18]. Supported by many examples such as these, MHS algorithms have been developed since the 1970s. These algorithms are mainly used to find global solutions in optimization problems. But finding these solutions with absolute certainty is a difficult task. Instead, it is accepted to find the closest solution to this global solution. MHS algorithms need two things to perform these search tasks. The first is exploitation and the second is exploration [19,20,21]. Although different Meta-Heuristic Algorithms have distinctive features within themselves, basically these two stages are common to all MHS algorithms. It defines the search of the solution space as wide, global and random as possible during the exploration phase. Exploitation phase defines the ability of the solution candidates sought in the exploration phase to search with high precision. If the algorithm's exploration ability is high, the randomness and diversity of the algorithm increases. If the algorithm's exploitation feature is more dominant, it performs more local search processes to increase the precision and quality of the solution candidates. Since each optimization problem has different features, these two features, exploitation and exploration, must complement each other in perfect balance. However, no MHS can definitively find the global optimum for all optimization problems [22]. This process is logically proven by the No Free Lunch (NFL) theory [23].

Thanks to the above-mentioned theorem, the researchers were motivated to design a large number of new MHSs. This is the source of motivation for our work. Thanks to the method we developed in our study, the performance of the search process increases, thus reducing the possibility of being caught in local minimum traps. For this, we propose the FDB-based SMA algorithm to more effectively determine the values that guide the solution candidates in the search process life cycle in MHS algorithms. Thanks to the proposed method, the SMA avoids local optima more effectively and does not converge prematurely. Many cases have been designed in applying the FDB method to the SMA algorithm. In order to compare the developed FDB-SMA algorithm and the base algorithm, 10 unconstrained problems in the CEC 2020 problem pool were designed and used in 30, 50 and 100 dimensions. In this way, the early convergence problem in the SMA algorithm has been eliminated by applying the FDB method. The local search and diversity capabilities of the algorithm have been improved. As a result, a powerful FDB-RUN algorithm that can be used in solving different types of optimization problems has been brought to the literature.

II. METHOD

Stochastic optimization algorithms are able to produce different results by taking the same input, thanks to the randomness feature. These algorithms are divided into two categories, heuristics and meta-heuristics. In these two categories, it actually tries to find the optimum value of the problem by guided trial and error [24]. While heuristics depend on the problem, meta-heuristics are called black boxes and have no prior assumptions about the problem they are trying to solve [2]. Each MHS algorithm is similar in terms of operation. This process consists of several successive steps. To mention these steps; It starts with defining the problem to be optimized first. Then, the problem parameters are determined and a community of solution candidates is created. Then, the fitness values of the candidate solutions are calculated and the search process life cycle is started. The search process includes the life cycle selection process, neighborhood search and diversity, and updating the solution candidates. This step is repeated until the optimum result is found, and then this process is terminated. The parts of MHS algorithms that make a difference in themselves emerge in the search process lifecycle. The first step of the mentioned process begins with the selection of candidates who will lead the search process. Three methods are used in the selection of these candidates: random, greedy and probabilistic. While the random selection method is used to provide diversity, the greedy selection method chooses those with the best compatibility value among the solution candidates. The probabilistic method is a mix of the first two methods. It has two techniques known as roulette and double tournament. In the roulette wheel technique, the pieces of the wheel are adjusted according to their fitness values, and the candidate represented by that piece is selected. In the double tournament technique, two randomly selected values from the population are compared and the one with the higher fitness value is taken [25].

A. SLİME-MOULD-ALGORİTHM

The slime mold algorithm was first proposed by Schmickland Crailsheim as a biology-inspired pathfinding principle for use in swarm robotics [26]. This proposed slimy mold algorithm simulates the foraging process of Physarum polycephalum by providing low-cost and minimally erroneous graphics to simulate. Derived from this living organism, this algorithm has been used in areas such as graph theory, production networks, and used in the field of graph optimization [27]. This developed SMA algorithm imitates the foraging cycle of slimy mold by giving positive or negative answers by using weight values in its algorithm.

The slime mold inspired by the SMA algorithm is actually a fungus, and the life cycle of this fungus was first published by Howard in 1931 in an article called slimy mold [28]. The main feeding process of this slimy mold living in cold and damp places is Plasmodium. The feeding process of this living organism first begins with the search for food, and after the food is found, the food is wrapped around it and continues with the secretion of enzymes to digest this food. In order to move towards the found food, the end of the slimy mold organism on the food side opens like a fan and connects the food to itself with venous networks extending towards the food. This organism, which can form a large number of these networks that bind the food to itself, can grow to more than 900 square centimeters depending on the amount of food around. As mentioned earlier, this organism creates the most appropriate way to connect to the nutrients around it, thanks to positive and negative feedback. Therefore, as mentioned before, the mathematical model of slime mold has been used in subjects such as graph theory and production networks [29,30].

This organism makes a choice according to the nutrient concentration when choosing various nutrients in its environment, and the thickness of the venous networks they use in binding to the food is directly proportional to the amount of cytoplasmic fluid flowing from the food to which they are attached to the organism [31,33]. This organism can split its biomass in half if it has found nutrients of equivalent quality [31].

If the concentration of nutrient sources around the slime mold organism is low, it will leave its environment in search of highly concentrated nutrients. This shows us that the organism's foraging patterns are dynamic according to food quality [33,34].
The foraging process of the slimy mold organism is mathematically modeled in two stages. These are Approach food, Wrap food stages.

A.1. Approach Food

In the mathematical model of the slimy mold organism's approach to the food, it is observed that it approaches the high concentration food it finds by contraction. The mathematical model of this is given below [22].

$$\overrightarrow{X(t+1)} = \begin{cases} \overrightarrow{X_b(t)} + \overrightarrow{vb}.(\overrightarrow{W}.\overrightarrow{X_A(t)} - \overrightarrow{X_B(t)}), & r < p \\ \overrightarrow{vc}.\overrightarrow{X(t)}, & r \geq p \end{cases} \quad (1)$$

Where $\overrightarrow{vb}$ [-a,a] is a parameter with a range. $\overrightarrow{vc}$ is a parameter that decreases linearly from one to zero. t is the number of iterations. $\overrightarrow{X_b}$ is the individual location with the highest odor concentration currently available. $\vec{X}$ is location of slimy mold, $\overrightarrow{X_A}$ ve $\overrightarrow{X_B}$ are two people randomly selected from the herd. $\overrightarrow{W}$ represents the weight of the slimy mold. The formula for p is given below [22].

$$p = tanh|S(i) - DF| \quad (2)$$

i∈1,2,…..,n, S(i) It is the suitability of $\vec{X}$. DF is the fitness value at all iterations..
The formula for $\overrightarrow{vb}$ is given below [22].

$$\overrightarrow{W(SmellIndex(\imath))} = \begin{cases} 1 + r \cdot log\left(\frac{bF - S(i)}{bF - wF} + 1\right), condition \\ 1 - r \cdot log\left(\frac{bF - S(i)}{bF - wF} + 1\right), \quad others \end{cases} \quad (3)$$

Where *S(i)* indicates that it is in the first half of the population, *r* indicates the random value in the range [0,1], *bF* indicates the optimal fit obtained in the current iterative process, and *wF* indicates the worst fit obtained.

A.2. Wrap Food

The location update formula of the slimy mold organism is given below [22].

$$\overrightarrow{X^*} = \begin{cases} rand \cdot (UB - LB) + LB, rand < z \\ \overrightarrow{X_b(t)} + \overrightarrow{vb} \cdot \left(W \cdot \overrightarrow{X_A(t)} - \overrightarrow{X_B(t)}\right), r < p \\ \overrightarrow{vc} \cdot \overrightarrow{X(t)}, r \geq p \end{cases} \quad (4)$$

LB and *UB* indicate the lower and upper limits of the search range, while rand and *r* indicate the random value in [0,1].

B. IMPROVED SLİME-MOULD-ALGORİTHM WİTH FİTNESS-DİSTANCE BALANCE BASED GUİDİNG MECHANİSM

The most important and fundamental steps in the meta-heuristic search process are the exploitation and exploration steps. Search operators throughout the MHS process are to carry out these two steps in the most effective way. To briefly mention these steps; From the population created in the first step, exploitation, solution candidates are selected according to their reference positions. In the second step, exploration, a search is made using the reference locations selected in the first step [35,36,37]. However, the diversity of most MHS algorithms in multidimensional and complex problems decreases, that is, early convergence occurs, and the solution candidates begin to resemble each other. In other words, the search process ends with a local optima trap [38,39,40]. In order to prevent this situation, although many MHS algorithms are encountered in the literature, these algorithms mostly focus on the development of search operators. However, the fact that the developed algorithms are not caught in the local optima trap is not only dependent on the improvement of search operators. What is essential for the success of this process is the joint and effective development of selection methods and search operators. For this purpose, it is aimed to overcome the local optima trapping problem by integrating the FDB method into the SMA algorithm [19,41,42]. With the equations given below, the application of the FDB method to the SMA algorithm is discussed with different cases.

Table 1. *Cases of SMA base algorithm created by integrating FDB method*

Base Algorithm	**Cases**

Eq(1) and Eq(2)	**CASE-1**
	if r<p
	*if(**rand<0.3**)*
	X(i,j) = bestPositions(j)+ vb(j)(weight(**fdbindex**,j)*X(A,j)-X(B,j));*
	else
	X(i,j) = bestPositions(j)+ vb(j)(weight(i,j)*X(A,j)-X(B,j));*
	end
	else
	*X(i,j) = vc(j)*X(**fdbindex**,j);*
	and
	*if(**rand<0.3**)*
	*p =tanh(abs(AllFitness(**fdbindex**)-Destination_fitness));*
	else
	p =tanh(abs(AllFitness(i)-Destination_fitness));
	end

Eq(1) and Eq(2)	**CASE-2**
	if r<p
	*if(**rand<0.5**)*
	X(i,j) = bestPositions(j)+ vb(j)(weight(i,j)*X(**fdbindex**,j)-X(fdbindex,j));*
	else
	X(i,j) = bestPositions(j)+ vb(j)(weight(i,j)*X(A,j)-X(B,j));*
	end
	else
	*X(i,j) = vc(j)*X(**fdbindex**,j);*
	and
	*if(**rand<0.5**)*
	*p =tanh(abs(AllFitness(**fdbindex**)-Destination_fitness));*
	else
	p =tanh(abs(AllFitness(i)-Destination_fitness));
	end

Eq(1) and Eq(2)	**CASE-3**
	if r<p
	*if(**rand<0.5**)*
	X(i,j) = bestPositions(j)+ vb(j)(weight(**fdbindex**,j)*X(**fdbindex**,j)-X(**fdbindex**,j));*
	Else
	X(i,j) = bestPositions(j)+ vb(j)(weight(i,j)*X(A,j)-X(B,j));*

Table 1 (continues). *Cases of SMA base algorithm created by integrating FDB method*

	End
	else
	*X(i,j) = vc(j)*X(**fdbindex**,j);*
	and
	*if(**rand<0.5**)*
	*p =tanh(abs(AllFitness(**fdbindex**)-Destination_fitness));*
	else
	p =tanh(abs(AllFitness(i)-Destination_fitness));
	end

Eq(1) and Eq(2)	**CASE-4**
	if r<p
	*if(**rand<0.5**)*
	X(i,j) = bestPositions(j)+ vb(j)(weight(i,j)*X(**fdbindex**,j)-X(**fdbindex**,j));*
	else
	X(i,j) = bestPositions(j)+ vb(j)(weight(i,j)*X(A,j)-X(B,j*
	end
	else
	*if(**rand<0.5**)*
	*X(i,j) = vc(j)*X(**fdbindex**,j);*
	else
	*X(i,j) = vc(j)*X(i,j);*
	end
	end
	and
	*if(**rand<0.5**)*
	*p =tanh(abs(AllFitness(**fdbindex**)-Destination_fitness));*
	else
	p =tanh(abs(AllFitness(i)-Destination_fitness));
	end

Eq(1) and Eq(2)	**CASE-5**
	if r<p
	*if(**rand<0.5**)*
	X(i,j) = bestPositions(j)+ vb(j)(weight(**fdbindex**,j)*X(**fdbindex**,j)-X(**fdbindex**,j));*
	else
	X(i,j) = bestPositions(j)+ vb(j)(weight(i,j)*X(A,j)-X(B,j));*
	end
	else
	*if(**rand<0.5**)*
	*X(i,j) = vc(j)*X(**fdbindex**,j);*
	else
	*X(i,j) = vc(j)*X(i,j);*
	end
	end
	and
	*if(**rand<0.5**)*
	*p =tanh(abs(AllFitness(**fdbindex**)-Destination_fitness));*
	else
	p =tanh(abs(AllFitness(i)-Destination_fitness));
	end

Table 1 (continues). *Cases of SMA base algorithm created by integrating FDB method*

	CASE-6

Eq(1)	if r<p
	if (rand<0.5)
	X(i,j) = bestPositions(j)+ vb(j)*(weight(i,j)*X(fdbindex,j)-X(fdbindex,j));
	else
	X(i,j) = bestPositions(j)+ vb(j)*(weight(i,j)*X(A,j)-X(B,j));
	end
	else
	X(i,j) = vc(j)*X(i,j);
	End

	CASE-7

Eq(1)	*if r<p*
	*if (**rand<0.3**)*
	X(i,j) = bestPositions(j)+ vb(j)(weight(i,j)*X(**fdbindex**,j)-X(**fdbindex**,j));*
	else
	X(i,j) = bestPositions(j)+ vb(j)(weight(i,j)*X(A,j)-X(B,j));*
	end
	else
	*X(i,j) = vc(j)*X(i,j);*
	End

	CASE-8

Eq(1) and Eq(2)	*if r<p*
	X(i,j) = bestPositions(j)+ vb(j)(weight(i,j)*X(**fdbindex**,j)-X(B,j));*
	else
	*if(**rand<0.3**)*
	*X(i,j) = vc(j)*X(**fdbindex**,j);*
	else
	*X(i,j) = vc(j)*X(i,j);*
	end
	end
	and
	*if(**rand<0.7**)*
	*p =tanh(abs(AllFitness(**fdbindex**)-Destination_fitness));*
	else
	p =tanh(abs(AllFitness(i)-Destination_fitness));
	end

All cases given in Table-1 are new designs of SMA algorithm with FDB method. The effectiveness of these cases has been proven by certain test results and is shared below.

III. EXPERİMENTAL STUDY

A. SETTİNGS

Extensive experimental studies have been carried out to test the efficiency and performance of the above-mentioned cases of the proposed FDB-SMA algorithm. In order to clearly show the efficiency and performance of the cases created with the developed algorithm, the cases created in four different types of unconstrained comparison problems (Unimodal, Basic Multimodal, Hybrid and Composition) were tested with the SMA base algorithm in different sizes (30/50/100) search space. naturalized. The following procedures were followed in order to carry out the experimental studies in an objective and fair manner:

- The conditions defined at the CEC 2020 conference are referenced for the experimental work settings [43].
- In editing the parameters of the SMA algorithm, the settings given in its own work, ie population size, etc. taken as reference.
- In order to ensure equality of opportunity between the base algorithm and the created cases, the termination criterion is defined over the maximum number of evaluations of the objective function. This value is 10,000*d (d:problem size).
- Dynamically resizable CEC 2020 comparison functions are used to reveal the performance of the proposed method in low, medium and high dimensional search fields. In the study, 30, 50 and 100 dimensional problems were created.
- Experimental studies were carried out on MATLAB®R2018b, on INTEL CORE i7 7700HQ 2.80 GHz and 16 GB RAM and x64 based processor.

B. BENCHMARK PROBLEMS

In the experimental studies, 10 different problems with four different types were used without constraints. All of the problems are taken from the CEC 2020 comparison pools [43]. Table 2 below shows the problems and which features of these problems optimization algorithms can test.

Table 2. *Problem type and characteristics*

Problem type	Problem no	Characteristics
Bent Cigar Function	1	It is the type of problem used to test the local search performance of algorithms.
Shifted and Rotated Schwefel's Function	2	It is the type of problem used to test the global search (diversity) performance of algorithms.
Shifted and Rotated Lunacek bi-Rastrigin Function	3	
Expanded Rosenbrock's plus Griewangk's Function	4	
Hybrid Function 1,2,3	5,6,7	It is the type of problem used to determine the balanced search performance of algorithms.
Composition Function 1,2,3	8,9,10	It is the type of problem used to test the capabilities of algorithms in search spaces with high complexity.

IV. ANALYZE RESULTS

In this section, the results obtained from the experimental studies are compared with the SMA base algorithm and FDB-SMA Cases in terms of performance. The Friedman test is used to compare the performances of the created cases and the base algorithm among themselves. Wilcoxon test is used to compare each FDB-SMA case and the SMA base algorithm pairwise. Test results are shown in the following sections.

A. STATİSTİCAL ANALYSİS RESULTS

In this section, the statistical performance analysis of the FDB-SMA cases created in this study and the base SMA algorithm is performed using the Friedman test. In Table-3, the results of the SMA base algorithm and FDB-SMA cases performed in 30/50/100 dimensions are given.

Table 3. *Experimental results according to Friedman analysis method*

		CEC 2020								
		Base	**Case-1**	**Case-2**	**Case-3**	**Case-4**	**Case-5**	**Case-6**	**Case-7**	**Case-8**
SMA	**D =30**	5.376	5.262	4.529	4.881	4.962	5.019	4.852	5.048	5.071
	D=50	5.471	5.286	4.743	4.752	5.052	5.133	4.486	4.938	5.138
	D=100	6.190	2.462	4.790	4.857	5.167	5.124	5.329	5.533	5.548
	Mean	5.679	4.337	4.687	4.830	5.060	5.092	4.889	5.173	5.252

When the values given in Table-3 are examined, all cases of the developed FDB-SMA algorithm give better results than the base SMA algorithm. When these results are examined, all the cases given of the developed FDB-SMA algorithm are proof that it is more effective than the base SMA algorithm.
When Table-4 is examined, the results of the binary Wilcoxon comparisons of the developed FDB-SMA algorithm cases with the base SMA algorithm are given. As it is understood from these results, it is seen that the developed algorithm cases achieve better results in all bots, especially in high dimensions.

Table 4. *Pairwise comparison of experimental results according to Wilcoxon analysis method*

		CEC 2020							
		Case-1	**Case-2**	**Case-3**	**Case-4**	**Case-5**	**Case-6**	**Case-7**	**Case-8**
vs. SMA +/=/-	D =30	1/9/0	1/9/0	1/8/1	2/7/1	0/9/1	1/9/0	1/9/0	2/8/0
	D=50	1/9/0	3/5/2	4/5/1	3/6/1	2/7/1	4/5/1	2/8/0	1/8/1
	D=100	7/3/0	4/6/0	4/6/0	4/6/0	4/6/0	3/7/0	4/6/0	3/7/0

The Friedman analysis results of all given cases of the developed FDB-SMA algorithm are given in table-5.

Table 5. *Friedman analysis results of the problems handled according to different dimension types*

Problem Type	**Dimension**	**Base**	**Case-1**	**Case-2**	**Case-3**	**Case-4**	**Case-5**	**Case-6**	**Case-7**	**Case-8**
Unimodal	D=30	5.67	4.90	4.76	4.76	4.48	4.62	6.29	5.71	3.81
	D=50	5.33	6.43	3.71	4.52	5.43	4.86	4.24	4.71	5.76
	D=100	4.71	5.95	4.14	4.62	4.38	5.05	5.81	5.90	4.43
Basic Multimodal	D=30	5.48	4.97	4.70	5.24	5.76	4.54	4.54	4.60	5.17
	D=50	5.49	4.84	4.89	5.29	4.86	5.41	4.43	5.00	4.79
	D=100	6.19	2.33	4.94	4.86	4.95	5.24	5.60	5.56	5.33
Hybrid	D=30	5.67	6.08	4.89	4.27	3.95	4.86	4.62	5.38	5.29
	D=50	6.25	6.06	4.52	4.22	4.62	4.71	4.14	5.03	5.43
	D=100	6.83	2.89	4.17	4.60	4.95	4.56	4.89	5.89	6.22
Composition	D=30	4.89	4.86	3.92	5.17	5.33	5.79	4.92	4.94	5.17
	D=50	4.71	4.57	5.16	4.83	5.56	5.37	4.97	4.86	4.98
	D=100	6.05	1.00	5.48	5.19	5.86	5.60	5.33	5.03	5.46

When Table-5 is examined, the cases given of the developed FDB-SMA algorithm mostly achieved better results in 30, 50 and 100 dimensions than the base SMA algorithm. The developed cases were run 21 times and the results were obtained.

When Table-6 is examined, the average of the minimum values and standard deviation values that the developed FDB-SMA algorithm found during the search for the minimum values of the problems in the cases are shown.

Table 6. *The mean and standard deviation values of the CEC 2020 problems in the experimental studies used*

F	**D**	**Base**	**Case-1**	**Case-2**	**Case-6**
F1	**30**	8.75E+03 (6.96E+03)	7.05E+03 (6.82E+03) +	6.97E+03 (7.27E+03) +	1.15E+04 (7.24E+03) -
	50	1.16E+04 (1.11E+04)	1.61E+04 (1.15E+04) +	4.78E+03 (3.76E+03) +	6.16E+03 (6.11E+03) +
	100	1.04E+04 (1.13E+04)	1.61E+04 (1.15E+04) -	8.19E+03 (1.14E+04) +	2.04E+04 (2.10E+04) -
F2	**30**	3.14E+03 (7.42E+02)	3.08E+03 (4.06E+02) +	3.01E+03 (4.66E+02) +	2.78E+03 (4.17E+02) +
	50	6.13E+03 (7.18E+02)	5.48E+03 (7.03E+02) +	5.76E+03 (7.39E+02) +	5.25E+03 (8.38E+02) +
	100	1.34E+04 (1.15E+03)	5.48E+03 (7.03E+02) +	1.31E+04 (1.04E+03) +	1.35E+04 (1.03E+03) -
F3	**30**	1.42E+02 (2.32E+01)	1.29E+02 (3.50E+01) +	1.26E+02 (2.75E+01) +	1.28E+02 (2.43E+01) +
	50	2.72E+02 (4.23E+01)	2.70E+02 (3.39E+01) +	2.71E+02 (4.52E+01) +	2.59E+02 (2.43E+01) +
	100	9.11E+02 (1.08E+02)	2.70E+02 (3.79E+01) +	7.61E+02 (9.44E+01) +	8.11E+02 (1.25E+02) +
F4	**30**	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00) =	0.00E+00 (0.00E+00) =	0.00E+00 (0.00E+00) =
	50	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00) =	0.00E+00 (0.00E+00) =	0.00E+00 (0.00E+00) =
	100	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00) =	0.00E+00 (0.00E+00) =	0.00E+00 (0.00E+00) =
F5	**30**	2.73E+05 (1.38E+05)	3.18E+05 (1.58E+05) -	2.29E+05 (1.51E+05) +	1.56E+05 (1.03E+05) +
	50	6.11E+05 (2.72E+05)	6.80E+05 (3.29E+05) -	3.60E+05 (1.43E+05) +	3.48E+05 (1.92E+05) +
	100	2.36E+06 (9.58E+05)	6.80E+05 (3.29E+05) +	1.24E+06 (2.89E+05) +	1.45E+06 (3.96E+05) +
F6	**30**	3.95E+02 (1.50E-02)	3.36E+02 (1.00E+02) +	2.54E+02 (1.35E+02) +	3.59E+02 (1.16E+02) +
	50	1.00E+03 (2.46E+02)	1.06E+03 (3.14E+02) -	7.70E+02 (1.91E+02) +	8.27E+02 (1.82E+02) +
	100	3.02E+03 (5.33E+02)	1.06E+03 (3.14E+02) +	2.83E+03 (6.20E+02) +	2.68E+05 (4.90E+02) -
F7	**30**	1.49E+05 (4.14E+04)	1.81E+05 (6.17E+04) -	1.77E+05 (1.52E+05) -	1.60E+05 (1.31E+05) -
	50	3.86E+05 (1.26E+05)	3.65E+05 (1.92E+05) +	2.05E+05 (1.17E+05) +	2.14E+05 (1.10E+05) +
	100	1.24E+06 (6.05E+05)	3.65E+05 (1.92E+05) +	4.56E+05 (1.52E+05) +	5.24E+05 (2.18E+05) +
F8	**30**	3.09E+03 (9.03E+02)	3.38E+03 (9.06E+02) -	2.24E+03 (1.41E+03) +	2.53E+03 (1.61E+03) +
	50	6.57E+03 (9.10E+02)	6.49E+03 (1.12E+03) +	5.95E+03 (8.97E+02) +	6.02E+03 (7.34E+02) +
	100	1.51E+04 (1.24E+03)	6.49E+03 (1.12E+03) +	1.41E+04 (1.06E+03) +	1.42E+04 (1.21E+03) +
F9	**30**	5.26E+02 (2.77E+01)	5.21E+02 (2.35E+01) +	5.41E+02 (3.41E+01) -	5.29E+02 (2.55E+01) -
	50	6.88E+02 (4.04E+01)	6.94E+02 (4.01E+01) -	7.30E+02 (6.26E+01) -	7.65E+02 (9.04E+01) -
	100	1.40E+03 (7.21E+01)	6.94E+02 (4.01E+01) +	1.40E+03 (7.55E+01) =	1.37E+03 (6.27E+01) +
F10	30	3.89E+02 (7.91E+00)	3.87E+02 (9.36E-01) +	3.88E+02 (1.24E+01) +	3.90E+02 (9.31E+00) -
	50	5.22E+02 (2.76E+01)	5.25E+02 (2.96E+01) -	5.50E+02 (4.16E+01) -	5.28E+02 (4.52E+01) -
	100	7.85E+02 (6.69E+01)	5.25E+02 (2.96E+01) +	7.75E+02 (6.24E+01) +	7.77E+02 (5.02E+01) +

When Table-6 is examined, the best three cases of the developed FDB-SMA algorithm are given. When the results were examined, it was able to find lower values than the base SMA algorithm. This shows that the developed algorithm is more effective.

B. CONVERGENCE ANALYSİS RESULTS

Box-plot graphics for the developed algorithm cases are given below. The given graphs are considered for 30/50/100 dimensional problems. It evaluates the box-plot capabilities of the algorithm by presenting the best results in the experimental studies in the given box-plot charts. Below are box-plot plots with D=30 in Figure 1, D=50 in Figure 2, and D=100 in Figure 3.

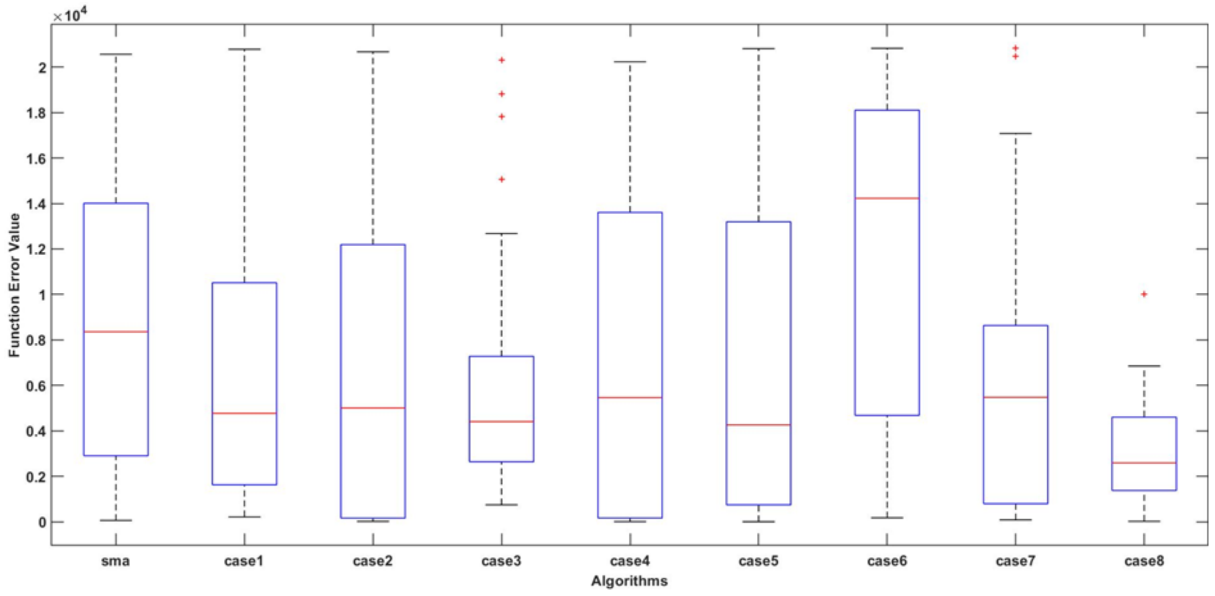

Fig 1. *Box-plot plots of SMA and FDB-SMA algorithms in 30 dimensions*

When Figure-1 is examined, almost all of the cases created in 30 dimensions have lower values than the base algorithm.

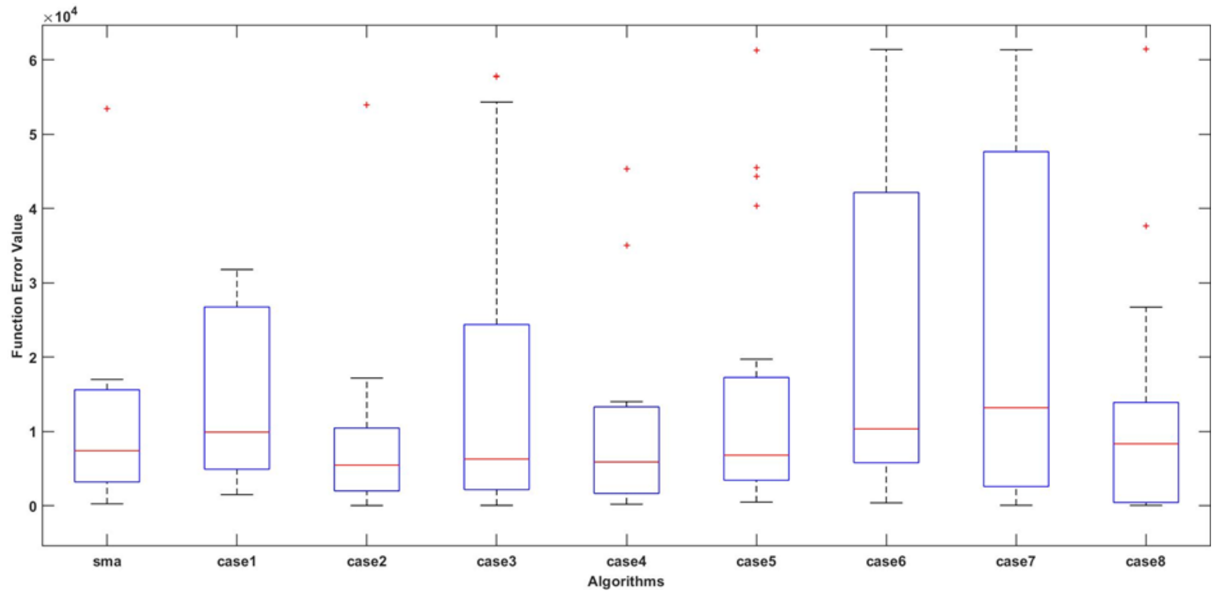

Fig 2. *Box-plot plots of SMA and FDB-SMA algorithms in 50 dimensions*

When Figure-2 is examined, all but one of the cases created in 50 dimensions achieved better results than the base algorithm.

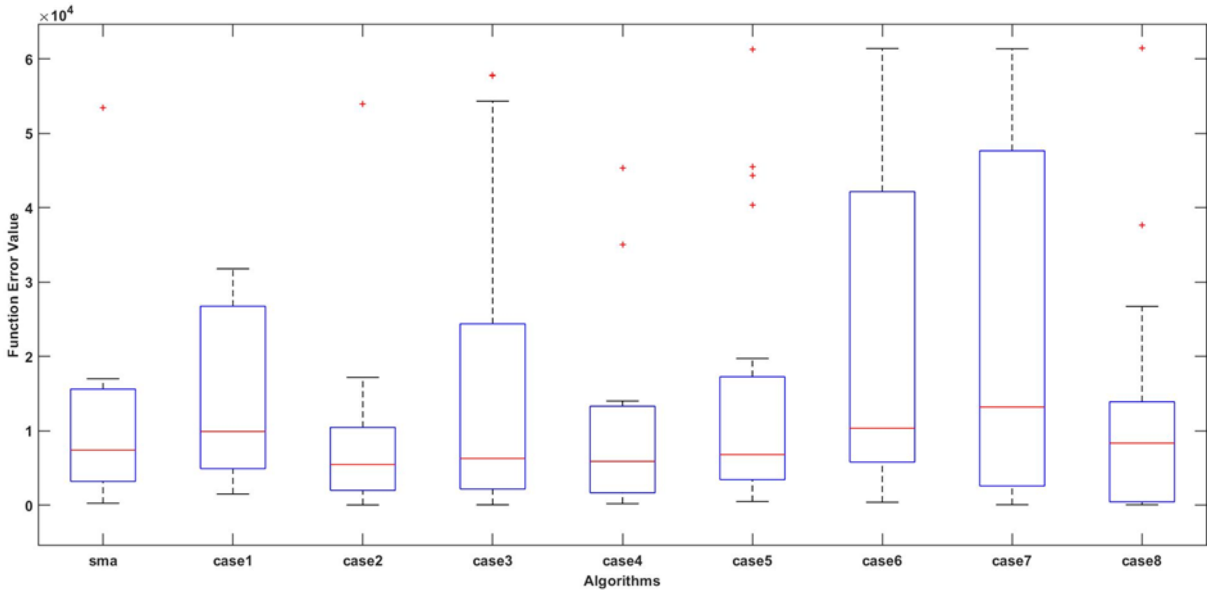

Fig 3. *Box-plot plots of SMA and FDB-SMA algorithms in 100 dimensions*

When Figure-3 is examined, relatively all of the cases created in 100 dimensions achieved better results than the base algorithm.

In Figure-1, the data distribution of all cases, except for two cases, has a wide distribution. It is observed in Figures-2 and 3 that there is a relatively narrow distribution of data.

V. CONLUSIONS

In this study, certain design changes have been made in the basic algorithm in order to improve the overall search performance of the SMA algorithm, which is a current meta-heuristic search algorithm. For this, a current selection method of Fitness-Distance Balance (FDB) was used. This method is effective in better identifying the solution candidates that guide the search process in meta-heuristic algorithms. In order to test the performance and effectiveness of the FDB-SMA algorithm developed in our study, it was subjected to certain experimental studies. These experimental studies were carried out within the framework of the rules and standards defined at the CEC conferences. In experimental studies, 10 unconstrained comparison problems taken from CEC 2020 were used in 30, 50 and 100 dimensions. The data obtained from the experimental studies were evaluated with statistical analysis methods such as Friedman and Wilcoxon, and it was seen that the FDB-SMA method obtained superior results than the base SMA algorithm. These results show that the FDB-SMA algorithm has achieved an effective improvement in diversity and convergence processes from the base algorithm and creates a delicate balance between neighborhood search diversity.

The MATLAB source codes of the FDB-RUN algorithm developed and proposed for the first time in this article will be shared on the MATLAB File Exchange platform after the article is published. You can search the MATLAB File Exchange platform with the keyword FDB-RUN to download the source codes.

VI. REFERENCES

[1] A. Kaveh and S. Talatahari, “An improved ant colony optimization for constrained engineering design problems,” *Engineering Computations*, vol. 27, no. 1, pp. 155-182, 2010.

[2] A. H. Halim, I. Ismail, and S. Das, “Performance assessment of the metaheuristic optimization algorithms: an exhaustive review,” *Artificial Intelligence Review*, vol. 54, no. 3, pp. 2323-2409, 2020.

[3] H. Chen, S. Jiao, A. A. Heidari, M. Wang, X. Chen and X. Zhao, “An opposition-based sine cosine approach with local search for parameter estimation of photovoltaic models,” *Energy Convers. Manage.*, vol. 195, pp. 927–942, 2019.

[4] H. Chen, Y. Xu, M. Wang and X. Zhao, “A balanced whale optimization algorithm for constrained engineering design problems,” *Appl. Math. Model.*, vol. 71, pp. 45–59, 2019.

[5] M. Wang, H. Chen, “Chaotic multi-swarm whale optimizer boosted support vector machine for medical diagnosis,” *Appl. Soft Comput.*, vol. 88, 2020.

[6] A. Kaveh and S. Talatahari, “A novel heuristic optimization method: charged system search,” *Acta Mechanica*, vol. 213, no. 3, pp. 267-289, 2010.

[7] L. J. Fogel, A. J. Owens, and M. J. Walsh, “Intelligent decision making through a simulation of evolution,” *Behavioral Science*, vol. 11, no. 4, pp. 253-272, 1966.

[8] D. E. Goldberg, and J. H. Holland, “Genetic Algorithms and Machine Learning,” *Machine Learning*, vol. 3, no. 2, pp. 95-99, 1988.

[9] F. Glover, “Heuristics for integer programming using surrogate constraints,” *Decision Sciences*, vol. 8, no. 1, pp.156-166, 1977.

[10] M. Drigo, “he Ant System: Optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol. 26, no. 1, pp. 1-13, 1996.

[11] R. Eberhart, and J. Kennedy, “A new optimizer using particle swarm theory,” in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39-43.

[12] D. Bertsimas, and J. Tsitsiklis, “Simulated annealing,” *Statistical Science*, vol. 8, no. 1, pp.10-15, 1993.

[13] A. Franzin and T. Stützle, "Revisiting simulated annealing: A component-based analysis," *Computers & Operations Research*, vol. 104, pp. 191-206, 2019.

[14] O. K. Erol, and I. Eksin, “A new optimization method: big bang–big crunch,” *Advances in Engineering Software*, vol. 37, no. 2, pp. 106-111, 2006.

[15] A. Kaveh and S. Talatahari, “Size optimization of space trusses using Big Bang–Big Crunch algorithm,” *Computers & Structures*, vol. 87, no. 17-18, pp. 1129-1140, 2009.

[16] E. Rashedi, H. Nezamabadi-Pour and S. Saryazdi, “GSA: a gravitational search algorithm,” *Information Sciences*, vol. 179, no. 13, pp. 2232-2248, 2009.

[17] Z. W. Geem, J. H. Kim, and G. V. Loganathan. “A new heuristic optimization algorithm: harmony search,” *Simulation*, vol. 76, no. 2 pp. 60-68, 2001.

[18] D. Karaboga, and B. Basturk, “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm,” *Journal of Global Optimization*, vol. 39, no. 3, pp. 459-471, 2007.

[19] H. T. Kahraman, S. Aras, and E. Gedikli, “Fitness-distance balance (FDB): a new selection method for meta-heuristic search algorithms,” *Knowledge-Based Systems*, vol. 190, 2020.

[20] R. Salgotra, U. Singh, and S. Saha, “New cuckoo search algorithms with enhanced exploration and exploitation properties,” *Expert Systems with Applications*, vol. 95, pp. 384-420, 2018.

[21] S. Aras, E. Gedikli, and H. T. Kahraman, “A novel stochastic fractal search algorithm with fitness-Distance balance for global numerical optimization,” *Swarm and Evolutionary Computation*, vol. 61, 2021.

[22] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, “Slime mould algorithm: A new method for stochastic optimization,” *Future Generation Computer Systems*, vol. 111, pp. 300-323, 2020.

[23] D. H. Wolpert, and W. G. Macready, “No free lunch theorems for optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no .1, pp. 67-82, 1997.

[24] X. S. Yang, *Nature-inspired Metaheuristic Algorithms*, 2nd ed., vol. 1, Cambridge, UK: Luniver Press, 2010, pp. 1-11

[25] M. Katı ve H. T. Kahraman, “Arz-talep tabanlı optimizasyon algoritmasinın fdb yöntemi ile iyileştirilmesi: mühendislik tasarım problemleri üzerine kapsamlı bir araştırma,” *Mühendislik Bilimleri ve Tasarım Dergisi*, c. 8, s. 5, ss. 156-172, 2020.

[26] T. Schmickl, and K. Crailsheim, “A navigation algorithm for swarm robotics inspired by slime mold aggregation,” in *International Workshop on Swarm Robotics*, Rome, Italy, 2006, pp.1-13.

[27] A. Brabazon, and S. McGarraghy, “Slime mould foraging: an inspiration for algorithmic design,” *International Journal of Innovative Computing and Applications*, vol. 11, no. 1, pp. 30-45, 2020.

[28] F. L. Howard, “The life history of Physarum polycephalum.” *American Journal of Botany*, vol. 18, no. 2, pp. 116-133, 1931.

[29] M. Becker, “On the efficiency of nature-inspired algorithms for generation of fault-tolerant graphs,” in *Proceedings - 2015 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2015*, 2016, pp. 1657-1663.

[30] V. ŠešumČavić, E. Kühn and D. Kanev, “Bio-inspired search algorithms for unstructured P2P overlay networks,” *Swarm Evol. Comput*, vol. 29, pp. 73–93, 2016.

[31] M. Beekman and Tanya Latty, “Brainless but multi-headed: decision making by the acellular slime mould Physarum polycephalum,” *Journal of Molecular Biology*, vol. 427, no. 23, pp. 3734-3743, 2015.

[32] T. Latty, and M. Beekman, “Speed–accuracy trade-offs during foraging decisions in the acellular slime mould Physarum polycephalum,” *Proceedings of The Royal Society B: Biological Sciences*, vol. 278, no. 1705, pp. 539-545, 2011.

[33] P. Kareiva and G. Odell, “Swarms of predators exhibit” preytaxis “if individual predators use area-restricted search,” *The American Naturalist*, vol. 130, no. 2, pp. 233-270, 1987.

[34] T. Latty and M. Beekman, “Food quality affects search strategy in the acellular slime mould, Physarum polycephalum,” *Behavioral Ecology*, vol. 20, no. 6, pp. 1160-1167, 2009.

[35] A. P. Piotrowski and J. J. Napiorkowski, “Step-by-step improvement of JADE and SHADE-based algorithms: Success or failure?,” *Swarm And Evolutionary Computation*, vol. 43, pp. 88-108, 2018.

[36] A. W. Mohamed, A. A. Hadi and K. M. Jambi, “Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization,” *Swarm and Evolutionary Computation*, vol. 50, pp. 1-14, 2019.

[37] N. Di Cesare and M. Domaszewski, “A new hybrid topology optimization method based on I-PR-PSO and ESO. Application to continuum structural mechanics,” *Computers & Structures*, vol. 212, pp. 311-326, 2019.

[38] S. Torabi and F. Safi-Esfahani, “Improved raven roosting optimization algorithm (IRRO),” *Swarm and Evolutionary Computation*, vol. 40, pp. 144-154, 2018.

[39] R. Cheng and Y. Jin, “A competitive swarm optimizer for large scale optimization,” *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191-204, 2014.

[40] X. Han, Q. Liu, H. Wang and L. Wang, “Novel fruit fly optimization algorithm with trend search and co-evolution,” *Knowledge-Based Systems*, vol. 141, pp. 1-17, 2018.

[41] U. Guvenc, S. Duman, H. T. Kahraman, S. Aras and M. Katı, “Fitness–Distance Balance based adaptive guided differential evolution algorithm for security-constrained optimal power flow problem incorporating renewable energy sources,” *Applied Soft Computing*, vol. 108, pp. 1-35, 2021.

[42] H.T. Kahraman, H. Bakir, S. Duman, M. Katı, S. Aras and U. Guvenc, “Dynamic FDB selection method and its application: modeling and optimizing of directional overcurrent relays coordination,” *Applied Intelligence*, pp. 1-36, 2021

[43] T. Kadavy, M. Pluhacek, A. Viktorin, and R. Senkerik, “SOMA-CL for competition on single objective bound constrained numerical optimization benchmark: a competition entry on single objective bound constrained numerical optimization at the genetic and evolutionary computation conference (GECCO) 2020,” in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 2020, pp. 9-10.