Düzce University
Journal of Science & Technology

Research Article

Improved Runge Kutta Optimizer with Fitness Distance Balance-Based Guiding Mechanism for Global Optimization of High-Dimensional Problems[1]

Enes CENGİZ [a,*], Cemal YILMAZ [b], Hamdi Tolga KAHRAMAN [c], Çağrı SUİÇMEZ [d]

[a] *Mechatronics Engineering, Technology Faculty, Afyon Kocatepe University, Afyonkarahisar, TURKEY*
[b] *Mingachevir State University, Mingachevir, AZERBAIJAN*
[c] *Software Engineering, Of Technology Faculty, Karadeniz Technical University, Trabzon, TURKEY*
[d] *Electrical Electronics Engineering, Technology Faculty, Gazi University, Ankara, TURKEY*
** Corresponding author's e-mail address: enescengiz@aku.edu.tr*
DOI: 10.29130/dubited.1014947

ABSTRACT

Runge Kutta (RUN) is an up-to-date and well-founded metaheuristic algorithm. The RUN algorithm aims to find the global best in solving problems by going beyond the traps of metaphors. For this purpose, enhanced solution quality mechanism is used to avoid local optimum solutions and increase the convergence speed. Although the RUN algorithm offers promising solutions, it is seen that this algorithm has shortcomings, especially in solving high dimensional multimodal problems. In this study, the solution candidates that guide the search process in the RUN algorithm are developed using the Fitness-Distance Balance (FDB) method. Thus, using the FDB-based RUN algorithm, the global optimum value of many optimization problems will be obtained in the future. CEC 2020 which has current benchmark problems was used to test the performance of the developed FDB-RUN algorithm. 10 different unconstrained benchmark problems taken from CEC 2020 were designed by arranging them in 30/50/100 dimensions. Experimental studies were carried out using the designed benchmark problems and analyzed with Friedman and Wilcoxon statistical test methods. According to the results of the analysis, it was seen that the FDB-RUN variations showed a superior performance compared to the base algorithm (RUN) in all experimental studies. In particular, it has been shown to provide more effective results for the continuous optimization of high-dimensional problems.

Keywords: *Meta-heuristic search, runge kutta algorithm, fitness-distance balance (FDB), benchmark problems*

Yüksek Boyutlu Problemlerin Global Optimizasyonu için Uygunluk Mesafe Dengesi Tabanlı Rehber Mekanizmasıyla Runge Kutta Optimize Edicinin İyileştirilmesi

ÖZ

Runge Kutta (RUN), güncel ve sağlam temellere sahip bir metasezgisel algoritmadır. RUN algoritması, metaforların tuzaklarının ötesine geçerek problemlerin çözümünde küresel en iyiyi bulmayı amaçlar. Bu amaçla, yerel optimum çözümlerden kaçınmak ve yakınsama hızını artırmak için geliştirilmiş çözüm kalitesi mekanizması kullanılmaktadır. RUN algoritması umut verici çözümler sunsa da bu algoritmanın özellikle yüksek boyutlu multimodal problemlerin çözümünde eksiklikleri olduğu görülmektedir. Bu çalışmada, Uygunluk-Mesafe Dengesi (FDB) yöntemi kullanılarak RUN algoritmasında arama sürecine rehberlik eden çözüm adayları geliştirilmiştir. Böylece FDB tabanlı RUN algoritması kullanılarak gelecekte birçok optimizasyon probleminin global optimum değeri elde edilecektir. Geliştirilen FDB-RUN algoritmasının performansını test etmek için güncel benchmark sorunları olan CEC 2020 kullanılmıştır. CEC 2020'den alınan 10 farklı kısıtsız kıyaslama problemi 30/50/100

[1] This study was presented in ICAIAME 2021 and published as summary text.

Received: 28/10/2021, Revised: 25/11/2021, Accepted: 30/11/2021

boyutlarında düzenlenerek tasarlanmıştır. Deneysel çalışmalar tasarlanan kıyaslama problemleri kullanılarak gerçekleştirilmiş ve Friedman ve Wilcoxon istatistiksel test yöntemleri ile analiz edilmiştir. Analiz sonuçlarına göre FDB-RUN varyasyonlarının tüm deneysel çalışmalarda temel algoritmaya (RUN) göre daha üstün bir performans gösterdiği görülmüştür. Özellikle yüksek boyutlu problemlerin sürekli optimizasyonu için daha etkili sonuçlar sağladığı gösterilmiştir.

Anahtar Kelimeler: *Meta-sezgisel arama, Runge Kutta algoritması, uygunluk-mesafe dengesi (FDB), kıyaslama problemleri*

I. INTRODUCTION

Stochastic search and optimization algorithms use different ways to reach the best solution. These algorithms are divided into two as Heuristic and Meta-Heuristic Search (MHS). While heuristic algorithms are designed for the problem, MHS's are designed independently of the problem. Many MHS algorithms have been developed to solve optimization problems. When these problems are handled with MHS algorithms, they are optimized using a nature-based heuristic method. These algorithms do not promise to offer the best solution to the problem due to their stochastic nature. However, these approaches can produce near-optimal results [1,2].

In the literature, various optimization algorithms have been developed, inspired by many fields such as physics, biology, mathematics, music, swarm and social-based. These algorithms imitate the real-life behaviour of the source they are inspired by. For example, an important algorithm in the field of optimization was introduced to the literature with the Genetic Algorithm (GA) in the 1960s and 1970s [3]. In the GA approach, inspired by the theory of evolution, the survival of the genes of living things in population is simulated [4]. In the 1980s, the Simulated Annealing (SA) algorithm was developed by being influenced by the field of physics. This optimization algorithm was inspired by the annealing process of metals [5,6]. In addition, the Tabu Search algorithm, which enables the use of memory in MHS algorithms, has been proposed in these years. Thus, while the algorithm is running, the initial information is kept in memory and ensured that it is not repeated in the next steps of the optimization process [7]. The ant colony algorithm (ACO) was developed by taking advantage of the swarm intelligence of social ants in the 90s. This algorithm, which was about determining the shortest distance between two points when it was first developed, is very popular in swarm algorithms [8]. In addition, particle swarm optimization (PSO) was developed in the same years, inspired by swarm behaviour. This algorithm, which is a population-based optimization technique, was created based on the social behaviour of bird flocks [9,10]. In the early 2000s, the Harmony Search (HS) optimization algorithm was developed, inspired by the field of music. This algorithm is inspired by the harmonious playing of instruments so that the harmony of the sound while performing the music is pleasing to the listener [11]. Artificial Bee Colony (ABC) algorithm was developed in 2007 by modelling the foraging process of bees [12]. In 2009, Gravitational Search (GS) [13] and Cuckoo Search (CS) [14] algorithms were introduced to the literature.

As a result of the studies carried out in the field of MHS algorithms in the last decade, many new algorithms have been developed. The quality of MHS algorithms depends on the extent of their exploration and exploitation capabilities. Most researchers consider efficiency and effectiveness measures for the performance of the MHS algorithm. Efficiency measures refer to the quality of the solution and the speed of reaching this solution [15,16]. Effectiveness measures reveal the capacity of the stochastic algorithm with comparative statistical analyses to determine the importance of the local and global optimum results of the solutions obtained [17]. Experimental study criteria for the effective evaluation of the performance of competing algorithms for the solution of different types of optimization problems have been organized at the Congress on Evolutionary Computation (CEC) conferences held as international level [32]. The rules that MHS algorithms must comply with have been determined with

these regulations made in 2014. Thus, a fairer comparison environment has been prepared for evaluation with statistical analysis methods [34].

Currently, in the studies carried out in the field of MHS algorithms, it is seen that the shortcomings of the existing algorithms are mostly improved. The fact that the search process is very powerful while determining the global optimum value of complex problems keeps the performance of the algorithm high. Conversely, if the search process gets stuck on local minimum values, this algorithm does not give good results. There are many studies in the literature where the performance of the search process has been improved or redesigned [18-22]. Successful application of inspiration from nature should be involved in improving or redesigning MHS algorithms. The search process in MHS algorithms depends on exploration and exploitation. The effective realization of these search processes will ensure that the global optimum is found in the solution of the problems. The Fitness-Distance Balance (FDB) selection method has been developed to determine the vectors that guide the solution candidates in the search process life cycle in MHS algorithms [23]. Solution candidates that are thought to contribute more to the solution in order to improve the search process are determined with this developed method. Powerful hybrid metaheuristic search algorithms have been developed using the FDB selection method. A few of these are Fitness Distance Balance-based Adaptive Guided Differential Evolution (FDBAGDE) [24], Lévy flight and FDB-based coyote optimization algorithm (LRFDBCOA) [25], Fitness Distance Balance-based Symbiotic Organism Search (FDBSOS) [23] and Fitness Distance Balance-based Stochastic Fractal Search (FDBSFS) [26] Fitness Distance Balance-based Supply-Demand- Optimizer (FDBSDO) [27].

This paper is a scientific study on the improvement of the search process performance of Runge Kutta (RUN) [28], which is a current optimization algorithm, with the FDB method. The RUN algorithm, like other MHSs, aims to find the global best in solving optimization problems by going beyond the pitfalls of metaphors. For this purpose, "Enhanced Solution Quality Mechanism" is used in the RUN algorithm to avoid local optimum solutions and to increase the convergence speed [28]. Although the RUN algorithm offers promising solutions in optimization problems, it is seen that this algorithm has shortcomings, especially in the solution of multimodal problems. This problem is due to the fact that most MSH algorithms get caught in the local minimum traps and converge prematurely. In order to overcome this, it is aimed to carry out the search process of the RUN algorithm more effectively by using the FDB method, which was introduced to the literature by Kahraman et al [23]. Initially, the search process of the RUN algorithm was examined and analysed. Necessary studies have been carried out for the complete application of the FDB method. Many variations are presented by integrating the FDB method into the RUN algorithm. 10 test functions from the CEC 2020 comparison problems were used to compare the performance of the developed FDB-RUN algorithm variations with the basic algorithm (RUN). These problems are used in 30, 50 and 100 dimensions. After the experimental study, the results were compared using Wilcoxon and Friedman statistical analysis methods [29, 30]. The analysis results show that the FDB method makes a positive contribution to the search process performance of the RUN algorithm. Thus, the early convergence problem in the RUN algorithm has been eliminated to a significant extent by integrating the FDB method. The algorithm's exploration and exploitation capabilities have been improved. Therefore, a powerful FDB-RUN algorithm that can be used in solving different types of optimization problems has been brought to the literature.
The original contributions of this study to the literature can be listed as follows;

- Delivering effective results in high dimensions
- High performance in current comparison problems.
- A powerful and performance-enhanced RUN variant is proposed.

The rest of the article is designed as follows: In Chapter 2, the developed RUN algorithm and the applied FDB method were mentioned. In addition, developed cases are also shown in this section. The settings and comparison problems for the comparison of the developed cases and the base algorithm were explained in section 3. In Chapter 4, experimental study results and statistical analysis methods and comparison results were given in detail. In Chapter 5, the conclusion part was presented.

II. METHOD

MHS algorithms try to find global optimum values while solving a problem. These algorithms consist of certain stages in themselves. After the parameters of the optimization problems are determined, solution candidates are created and the fitness values of these solution candidates are calculated. Then, the search process lifecycle is run and the search process is started. In the last stage, the optimization process of the problem is completed after the necessary condition in the termination criterion is met. The flow chart of MHS algorithms is given in Figure 1 [33].

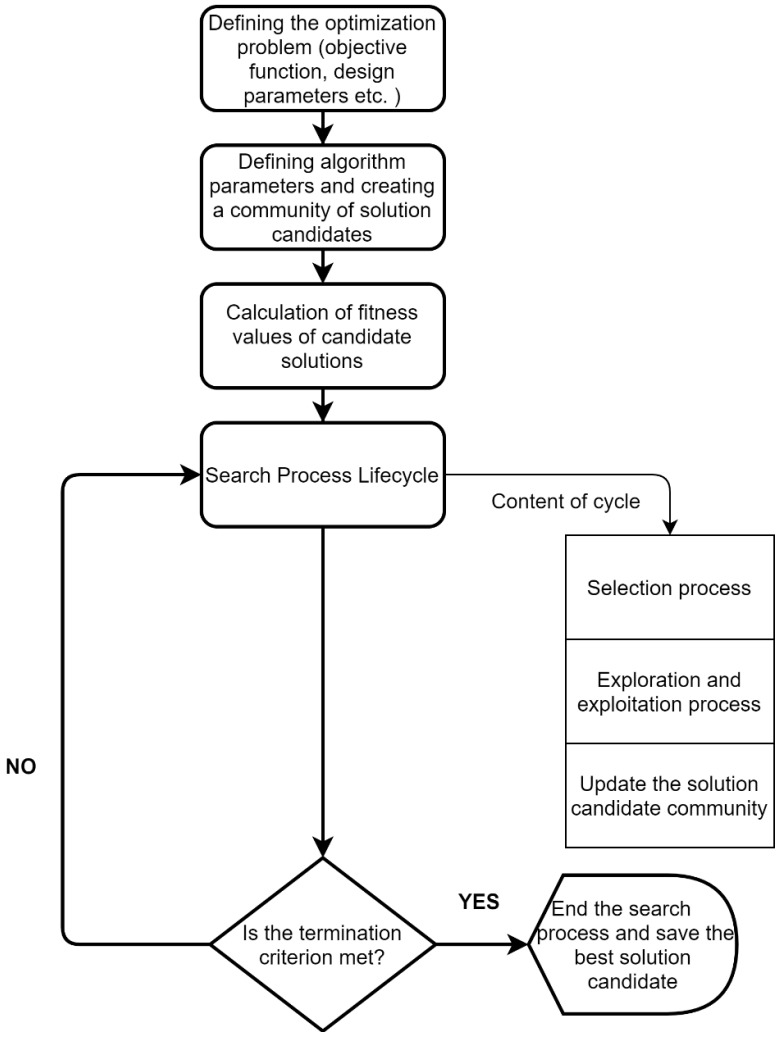

Figure 1. *General flow chart of MSH algorithms [33]*

The part before the Search process lifecycle stage in Figure 1 is common in MHS algorithms. The difference which is seen in the solution of problems in MHS algorithms occurs with the stage specified by Search process lifecycle. In short, the performance of algorithms mostly depends on the operations in this step.

Solution candidates are selected from this population since MHS algorithms are population-based. The first step in the search process is the selection of candidates who will guide this process. In the literature, there are different methods for selecting solution candidates from the population. The first of these are to determine the solution candidates randomly from the population. The second method is the selection of guides using the greedy solution selection method, taking into account the fitness values of the initial candidates [23, 26]. Another method is the selection of solution candidates using the probabilistic method known as roulette wheel and double tournament, again taking into account the fitness functions

[31]. Probability values are assigned proportionally according to the fitness function determined from the objective function of the candidates in the probabilistic solution method The roulette wheel is arranged according to these probability values and the randomly chosen part belongs to whichever candidate is chosen. The largest fitness value of two randomly selected candidates from the population is selected in the double tournament.

A. RUNGE-KUTTA METHOD

Bringing a new optimization algorithm to the literature is a very difficult process. Maintaining the balance between exploration and exploitation is one of the factors that affect the performance of optimization algorithms. In 2021, the Runge Kutta (RUN) optimization algorithm based on the Runge Kutta method was developed [28]. The RUN algorithm has been created in two structures to solve the problems with high performance. The first one is based on the Runge Kutta method to effectively complete the search process. Secondly, Enhanced Solution Quality (ESQ) structure is used to increase the solution quality. The RUN algorithm performs the process of updating the population in three different stages in the search process life cycle. Thus, the population of xn is updated as xn+1, xnew2 and xnew3. In the equations 1,2 and 3 below, the generation of xn+1, xnew2 and xnew3 populations is given, respectively [28].

if rand <0.5
(exploration phase)
$x_{n+1}=x_c + r \times SF \times g \times x_c) + SF \times SM + \mu \times x_s$
else (1)
(exploitation phase)
$x_{n+1}=x_m + r \times SF \times g \times x_c) + SF \times SM + \mu \times x_s,$
end

$x_s=randn\ (x_m - x_c)$ (2)

$x_s,= randn\ (x_{r2} - x_{r2})$ (3)

$x_m= \varphi \times x_n + (1 - \varphi) \times x_{r1}$ (4)

$x_c= \varphi \times x_{best} + (1 - \varphi) \times x_{lbest}$ (5)

Here φ is a random number in the range (0,1). $xbest$ represents the best solution in the population. $xlbest$ is the best position achieved at each iteration. In Equation 1, the r value is the parameter that takes the value of -1 or 1. The variable g can take values from 0 to 2. As the number of iterations increases, the local search around xc decreases. x_{n+1} shows the RUN search mechanism in the next iteration [28].

$SF = 2 \times (0.5 - rand) \times f$ (6)

$f = a \times exp(-b.rand \times (\frac{i}{Maxi})$ (7)

$\mu = 0.5 + 0.1 \times randn$ (8)

Here b and c are constants. i and Maxi are the number of iterations and the maximum number of iterations, respectively. SF has been used to strike a balance between exploration and exploitation [28].

$SM = 1/6 \times (xRK) \times \Delta x$ (9)

$x_{RK}= k1 + 2 \times k2 + 2 \times k3 + k4$ (10)

$k1 =\frac{1}{2\Delta x} \times (rand \times w - u \times xb)$ (11)

$u = round\ (1 + rand) \times (1 - rand)$ (12)

Here rand takes a value between 0 and 1. In the search process, the random parameter u is used to improve the best solution [28].

$\Delta x = 2 \times rand \times |Stp|$ (13)

$Stp = rand \times ((xb - rand \times xavg) + \gamma)$ (14)

$\gamma = rand \times (x_n - rand \times (u - l)) \times \exp(-4 \times \frac{i}{Maxi})$ (15)

Stp is the step size determined by the difference between xb and $xavg$. The γ parameter is an exponentially decreasing value. The $xavg$ parameter gives the mean value of all solutions in each iteration [28].

$$k2 = \frac{1}{2\Delta x}((rand \text{ x } (xw + rand1 \text{ x } k1 \text{ x } \Delta x) - (u \text{ x } xb + rand2 \text{ x } k1 \text{ x } \Delta x)) \quad (16)$$

$$k3 = \frac{1}{2\Delta x}(((rand \text{ x } (xw + rand1 \text{ x } (\frac{1}{2}k2) \text{ x } \Delta x) - (u \text{ x } xb + rand2 \text{ x } (\frac{1}{2}k2) \text{ x } \Delta x)) \quad (17)$$

$$k4 = \frac{1}{2\Delta x}(rand \text{ x } (xw + rand1 \text{ x } k3 \text{ x } \Delta x) - (u \text{ x } xb + rand2 \text{ x } k3 \text{ x } \Delta x)) \quad (18)$$

ESQ is used to improve the quality of the solution values in the RUN algorithm. With the proposed ESQ, three randomly taked solutions are averaged (x_{avg}), and then a new solution (xnew1) is generated with the best result (x_b). In addition, xnew2 solutions are obtained using x_{avg} and x_{new1} [28].

if rand <0.5
  if w <1
  $x_{new2} = x_{new1} + r \text{ x } w \mid (x_{new1} - x_{avg}) + randn \mid$
  Else
  $x_{new2} = (x_{new1} - x_{avg}) + r \text{ x } w \mid (x_{new1} - x_{avg}) + randn \mid$ (19)
  end
end

$$w = rand(0.2) \; x \; exp(-c(\frac{i}{Maxi}) \quad (20)$$

$$x_{avg} = \frac{x_{r1}+x_{r2}+x_{r3}}{3} \quad (21)$$

$$x_{new1} = x \; x_{avg} + (1-\beta) \; x \; x_{best} \quad (22)$$

Equations 20, 21, and 22 help determine xnew2 in equation 19. Here β is a random number between 0 and 1. R is a number that takes a value of 1, 0, or -1. x_best represents the best solution ever. c is equal to 5 x rand.
The x_{new2} solution values created in the search process lifecycle in the RUN optimization algorithm may not give better results than the current solution. In this case, x_{new3} solutions were created for new different solutions [28].

if rand <w
  $x_{new3} = (x_{new2} - rand \text{ x } x_{new2}) + SF \text{ x } (rand \text{ x } X_{RK} + (v \text{ x } x_b - x_{new2})$ (23)
end

Here v is a random number of $2 \times rand$. The main purpose of creating xnew 3 solutions is to try to find a better solution than xnew2 [28].

B. IMPROVED RUN WITH FITNESS-DISTANCE BALANCE BASED GUIDING MECHANISM

In the RUN algorithm, local traps occur while performing the search process life cycle. In this section, the development of the RUN optimization algorithm using the FDB method is explained. Thus, candidates who guide the search process will be better selected. In Table 1, the application of the FDB selection method to the algorithm is given to determine the best solution in the RUN algorithm. The 10 different cases created were handled one by one. In Table 1, the guidelines selected by applying the FDB method are shown with fdb.

Table 1. Cases of the base algorithm created using the FDB method

Base Algorithm	Cases
$Eq\ (21)$	**CASE-1** $x_{avg} = \{X(fdb,:) + X(fdb,:) + X(fdb,:)/3;\ rand < 0.9\ (X(A,:) + X(B,:) + X(C,:))/3;\ rand \geq 0.9$
$Eq\ (21)$	**CASE-2** $x_{avg} = \{X(fdb,:) + X(fdb,:) + X(fdb,:)/3;\ rand < 0.7\ (X(A,:) + X(B,:) + X(C,:))/3;\ rand \geq 0.7$
$Eq\ (21)$	**CASE-3** $Xavg = (X(fdb,:) + X(fdb,:) + X(fdb,:))/3;$
$Eq\ (21)$	**CASE-4** $Xavg = (X(fdb,:) + X(fdb,:) + X(C,:))/3;$
$Eq\ (21)$	**CASE-5** $x_{avg} = \{X(fdb,:) + X(fdb,:) + X(fdb,:)/3;\ rand < 0.4\ (X(A,:) + X(B,:) + X(C,:))/3;\ rand \geq 0.4$
$Eq\ (1)$ and $Eq\ (21)$	**CASE-6** $if\ rand < 0.5$ $Xnew = (Xc + r.* SF(i).* g.* Xc) + SF(i).* (SM) + mu.* (Xm - Xc);$ $else$ $Xnew = (Xm + r.* SF(i).* g.* Xm) + SF(i).* (SM) + mu.* (X(fdb,:) - X(B,:));$ End and $Xavg = (X(fdb,:) + X(B,:) + X(C,:))/3;$
$Eq\ (1)$ and $Eq\ (21)$	**CASE-7** $if\ rand < 0.5$ $Xnew = (Xc + r.* SF(i).* g.* Xc) + SF(i).* (SM) + mu.* (Xm - Xc);$ $else$ $Xnew = (Xm + r.* SF(i).* g.* Xm) + SF(i).* (SM) + mu.* (X(fdb,:) - X(B,:));$ end and $Xavg = (X(A,:) + X(fdb,:) + X(C,:))/3;$
$Eq\ (1)$ and $Eq\ (21)$	**CASE-8** $if\ rand < 0.5$ $Xnew = (Xc + r.* SF(i).* g.* Xc) + SF(i).* (SM) + mu.* (Xm - Xc);$ $else$ $Xnew = (Xm + r.* SF(i).* g.* Xm) + SF(i).* (SM) + mu.* (X(fdb,:) - X(B,:));$ end and $Xavg = (X(A,:) + X(B,:) + X(fdb,:))/3;$
(Eq.1), (Eq.11), (Eq.16), (Eq.17) and (Eq.18)	**CASE-9** $if\ rand < 0.5$ $Xnew = (Xc + r.* SF(i).* g.* Xc) + SF(i).* (SM) + mu.* (Xm - Xc);$ $else$ $Xnew = (Xm + r.* SF(i).* g.* Xm) + SF(i).* (SM) + mu.* (X(fdb,:) - X(B,:));$ end **fdb** is applied instead of xb in (Eq.1.10), (Eq.1.15), (Eq.1.16) and (Eq.1.17)
$Eq\ (1)$	**CASE-10** **if** rand<0.4 **if** rand<0.5 Xnew = (Xc+r.*SF(fdb).*g.*Xc) + SF(i).*(SM) + mu.*(Xm-Xc); **else** Xnew = (Xm+r.*SF(i).*g.*Xm) + SF(i).*(SM)+ mu.*(X(fdb,:)-X(B,:)); **end** **else** **if** rand<0.5 Xnew = (Xc+r.*SF(i).*g.*Xc) + SF(i).*(SM) + mu.*(Xm-Xc); **else** Xnew = (Xm+r.*SF(i).*g.*Xm) + SF(i).*(SM)+ mu.*(X(A,:)-X(B,:)); **end** **end**

Each variation given in Table 1 corresponds to new designs of the RUN algorithm. Whether these design changes are effective or not has been investigated with a comprehensive experimental study.

III. EXPERIMENTAL STUDY

A. SETTINGS

An extensive experimental study was carried out to test and verify the search performance of the proposed FDBRUN algorithm. The aim is to show the effect and performance of the cases developed using the FDB method. For this purpose, four different types of unconstrained comparison problems (Unimodal, Basic Multimodal, Hybrid and Composition) were also used in experimental studies. The performances of the base model of the RUN algorithm and the FDBRUN variations in search spaces of different sizes (30,50 and 100) are tested. Some procedures have been carried out to ensure that the experimental studies are objective and fair:

- The conditions defined at the CEC 2020 conference are referenced for the experimental operating settings [32].
- In the editing of the parameters of the RUN algorithm, the algorithm settings given in its study were 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 performed on MATLAB®R2019b, AMD Ryzen 7 4800H 2.90 GHz and 16 GB RAM and x64-based processor.

B. BENCHMARK PROBLEMS

Unconstrained 10 different optimization problems were used in experimental studies. All of the problems are taken from the CEC 2020 comparison pools [32]. Table 2 below gives the problems and their testing characteristics.

Table 2. *Test problems and features [32]*

Problem No	Problem Name	Characteristics
F1	Bent Cigar Function	This function is used to test the local search efficiency of algorithms.
F2	Shifted and Rotated Schwefel's Function	It is used to test the global search capabilities of algorithms.
F3	Shifted and Rotated Lunacek bi-Rastrigin Function	
F4	Expanded Rosenbrock's plus Griewangk's Function	
F5	Hybrid Function 1	It is used to determine the balanced search capabilities of algorithms.
F6	Hybrid Function 2	
F7	Hybrid Function 3	
F8	Composition Function 1	It is preferred for testing high complexity problems in search spaces.
F9	Composition Function 2	
F10	Composition Function 3	

In Table 2, 10 comparison problems taken from CEC 2020 and the features with which these problems are compared in optimization algorithms are given. F2 function tests whether the algorithm is stuck on local optimum points during the search process. F2, F3 and F4 functions show whether they are effectively provided in the search process. F5, F6 and F7 functions test whether it performs a balanced search during the search process. F8, F9 and F10 functions are used for performance testing in high search space [23, 24, 25, 26, 27].

IV. ANALYSIS RESULTS

In this section, a comparative performance analysis of RUN and FDBRUN cases (10 cases) is made by using the results obtained from the experimental studies. Friedman test was applied to compare the performances of the algorithms with each other. It se algorithm and 10 cases according to performance with this test. Wilcoxon test was applied to compare the base model RUN algorithm and FDBRUN cases as pairs. Accordingly, the results of the analysis are presented in the subsections, respectively.

A. STATISTICAL ANALYSIS RESULTS

After the experimental studies, performance analysis was performed with statistical analysis methods. Two different statistical analysis methods were used. The Friedman test performed using FDBRUN cases and the base algorithm (RUN) is presented in Table 3. Analysis results in 30, 50 and 100 dimensions are given.

Table 3. *Presentation of the experimental results according to the Friedman analysis method*

		CEC 2020										
		Base	**Case -1**	**Case -2**	**Case -3**	**Case -4**	**Case -5**	**Case -6**	**Case -7**	**Case -8**	**Case -9**	**Case -10**
RUN	**D =30**	7.667	6.439	6.494	6.824	6.680	6.820	6.792	7.196	7.041	7.751	7.316
	D=50	7.727	6.565	6.761	6.655	6.888	6.800	6.939	7.002	7.110	7.198	7.273
	D=100	8.463	6.575	6.429	6.951	6.971	6.922	6.975	6.596	6.816	6.980	7.412
	Mean	7.952	6.526	6.561	6.810	6.846	6.847	6.902	6.931	6.989	7.310	7.334

All of the developed FDBRUN cases gave better results than the base algorithm in the mean value of 30, 50 and 100 dimensions when the results given in Table 3 are examined. This result means that these FDB-based cases are better designed with the Runge Kutta approach.

Wilcoxon analysis method was used for pairwise comparison of FDBRUN cases developed with the base algorithm after the experimental study. All cases were compared in pairs with the base algorithm with this analysis. Pairwise comparison results according to Wilcoxon analysis method are given in Table 4.

Table 4. *Presenting the experimental results according to Wilcoxon analysis method as pairwise comparison*

		CEC 2020									
		Case -1	**Case -2**	**Case -3**	**Case -4**	**Case -5**	**Case -6**	**Case -7**	**Case -8**	**Case -9**	**Case -10**
vs.	**D =30**	3/7/0	3/7/0	3/7/0	3/5/2	3/6/1	3/6/1	3/7/0	4/3/3	3/7/0	1/6/3
RUN	**D=50**	2/8/0	4/6/0	2/8/0	2/7/1	2/8/0	2/8/0	3/6/1	3/6/1	2/8/0	3/5/2
+/=/-	**D=100**	4/6/0	4/6/0	4/6/0	4/5/1	4/6/0	5/5/0	4/6/0	4/6/0	4/6/0	4/6/0

As can be seen in Table 4, better results were obtained than the base algorithm (RUN) in all problem dimensions, except Case-9 (D=30), which is one of the cases created based on FDB. The performance of the algorithm is shown in Table 4. All of the developed cases give much better results than the base algorithm in solving large-sized problems according to Wilcoxon scores.

The results of the FDB-based cases according to the Friedman analysis are given in Table 5.

Table 5. *Results of test problems used in different dimensions according to Friedman analysis method*

Problem Type	Dim	Base	Case - 1	Case -2	Case -3	Case -4	Case -5	Case -6	Case -7	Case -8	Case -9	Case -10
Unimodal	D=30	7.08	6.82	6.86	7.59	7.33	6.57	6.88	6.33	6.37	7.28	6.26
	D=50	8.49	6.73	7.14	7.82	9.41	8.96	6.61	6.43	5.63	3.84	6.33
	D=100	8.18	7.49	7.37	6.98	9.12	8.28	7.00	5.51	6.49	3.73	6.39
Basic Multimodal	D=30	7.90	6.08	7.08	6.12	6.56	6.98	7.17	6.69	6.78	7.90	7.77
	D=50	8.33	6.35	6.36	6.08	6.63	6.80	6.80	7.18	6.81	7.99	7.56
	D=100	8.34	6.78	6.33	7.31	6.60	7.03	6.67	6.59	6.72	7.38	7.84
Hybrid	D=30	6.57	6.83	5.86	7.41	6.70	6.58	6.09	8.22	7.78	7.75	6.70
	D=50	6.89	6.48	6.65	7.22	6.44	6.41	6.26	7.54	8.28	8.07	6.43
	D=100	8.82	6.39	6.33	6.69	6.75	6.61	6.98	7.03	7.26	7.00	6.76
Composition	D=30	8.73	6.28	6.42	6.69	6.57	6.98	7.09	6.97	6.78	7.76	7.83
	D=50	7.71	6.81	7.14	6.28	6.75	6.47	7.88	6.48	6.73	6.65	8.14
	D=100	8.33	6.25	6.32	6.84	6.85	6.67	7.27	6.54	6.58	7.65	7.97

FDBRUN cases developed in most of the test problems gave better results than the RUN algorithm according to Table 5. When the 30, 50 and 100 dimensions are examined one by one, all the cases generally showed higher performance results. In addition, the developed Case-1 has demonstrated a more balanced search performance compared to its competitors in the search for global solutions in three different dimensions of four problem types. It was run 51 times to more accurately observe the performance of the developed cases. As a result of this run, the algorithms searched for the global minimum value of the test functions. The mean and standard deviation of the minimum values found after the completion of this optimization process are given in Table 6.

Table 6. *Mean and standard deviation values of CEC 2020 problems which are used in experimental work*

F	D	Base	Case-1	Case-2	Case-3
F1	30	3.63E+03 (4.15E+03)	4.47E+03 (4.69E+03)-	4.47E+03 (4.69E+03)-	4.76E+03 (5.20E+03)-
	50	1.05E+04 (3.69E+03)	9.77E+03 (4.84E+03)+	9.74E+03 (3.36E+03)+	1.05E+04 (3.69E+03)+
	100	4.71E+04 (1.53E+04)	4.56E+04 (2.02E+04)+	4.59E+04 (2.32E+04)+	4.28E+04 (1.74E+04)+
F2	30	2.97E+03 (5.32E+02)	3.19E+03 (6.73E+02)-	3.19E+03 (6.73E+02)-	3.23E+03 (7.98E+02)-
	50	6.66E+03 (1.95E+03)	6.54E+03 (1.48E+03)+	6.21E+03 (9.74E+02)+	6.66E+03 (1.95E+03)=
	100	1.47E+04 (2.08E+03)	1.56E+04 (4.79E+03)-	1.40E+04 (1.70E+03)+	1.69E+04 (5.30E+03)-
F3	30	3.16E+02 (7.01E+01)	1.95E+02 (4.50E+01)+	1.95E+02 (4.50E+01)+	2.31E+02 (5.36E+01)+
	50	4.63E+02 (6.80E+01)	4.66E+02 (9.48E+01)-	4.97E+02 (9.17E+01)-	4.63E+02 (6.80E+01)=
	100	1.93E+03 (2.22E+02)	1.60E+03 (1.97E+02)+	1.66E+03 (2.03E+02)+	1.61E+03 (1.92E+02)+

Table 6. *(continues) Mean and standard deviation values of CEC 2020 problems which are used in experimental work*

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	1.11E+04 (6.63E+03)	1.38E+04 (8.51E+03)-	1.38E+04 (8.51E+03)-	1.01E+04 (7.34E+03)+
	50	6.90E+04 (4.14E+04)	5.87E+04 (3.16E+04)+	7.32E+04 (4.90E+04)-	6.90E+04 (4.14E+04)=
	100	3.47E+05 (8.60E+04)	2.46E+05 (6.88E+04)+	2.48E+05 (6.20E+04)+	2.49E+05 (8.01E+04)+
F6	30	3.58E+02 (1.64E+02)	2.25E+02 (1.49E+02)+	2.25E+02 (1.49E+02)+	2.98E+02 (1.55E+02)+
	50	7.17E+02 (3.04E+02)	8.41E+02 (2.74E+02)+	9.17E+02 (3.03E+02)-	7.17E+02 (3.04E+02)=
	100	3.76E+03 (7.18E+02)	3.17E+03 (6.96E+02)+	3.19E+03 (6.34E+02)+	2.94E+03 (7.28E+02)+
F7	30	1.07E+04 (5.20E+03)	1.05E+04 (5.50E+03)+	1.05E+04 (5.50E+03)+	8.98E+03 (4.79E+03)+
	50	3.02E+04 (1.39E+04)	2.36E+04 (8.24E+03)+	2.27E+04 (6.78E+03)+	3.02E+04 (1.39E+04)=
	100	1.13E+05 (3.12E+04)	8.84E+04 (3.70E+04)+	8.72E+04 (3.40E+04)+	9.83E+04 (4.47E+04)+
F8	30	7.76E+02 (1.49E+03)	9.13E+02 (1.61E+03)-	9.13E+02 (1.61E+03)-	1.03E+03 (1.85E+03)-
	50	6.54E+03 (1.29E+03)	7.12E+03 (1.99E+03)-	6.67E+03 (1.69E+03)-	6.54E+03 (1.29E+03)=
	100	1.68E+04 (1.64E+03)	1.74E+04 (5.02E+03)-	1.72E+04 (4.00E+03)-	1.78E+04 (4.37E+03)-
F9	30	5.07E+02 (3.12E+01)	4.85E+02 (2.04E+01)+	4.85E+02 (2.04E+01)+	4.87E+02 (2.09E+01)+
	50	6.32E+02 (3.61E+01)	6.40E+02 (3.34E+01)-	6.53E+02 (3.75E+01)-	6.32E+02 (3.61E+01)=
	100	1.51E+03 (9.83E+01)	1.39E+03 (7.05E+01)+	1.39E+03 (7.31E+01)+	1.41E+03 (9.26E+01)+
F10	30	4.03E+02 (1.58E+01)	3.91E+02 (1.06E+01)+	3.91E+02 (1.06E+01)+	3.91E+02 (8.72E+00)+
	50	5.77E+02 (4.21E+01)	5.85E+02 (3.35E+01)-	5.85E+02 (3.34E+01)-	5.77E+02 (4.21E+01)=
	100	8.25E+02 (6.11E+01)	8.18E+02 (5.23E+01)+	8.11E+02 (4.87E+01)+	8.07E+02 (6.01E+01)+

In Table 6, the best 3 of the developed FDBRUN cases (Case-1, Case-2 and Case-3) and the results of the base algorithm are given. There are +, - and = signs next to the mean and standard deviations of the cases considered in the relevant table. These signs represent the comparison results of the cases according to the base algorithm. The developed cases mostly managed to find the minimum value compared to the base algorithm when the results in Table 6 are examined. Only Case-1, Case-2 and Case-3 in F7 function and Case-1 and Case-3 in F2 function showed lower performance than the base algorithm when the results of the problems in 100 dimensions were examined. In other results, it is seen that the developed cases outperform the base algorithm. This shows that the developed FDBRUN cases provide effective solutions in finding the global optimum value of high-dimensional problems.

In this section, box-plot figures of RUN and FDBRUN variations are presented. In order to examine the box-plot capabilities of the algorithms, the best results from the experimental studies were taken into account. From the selected problems, search spaces of 30/50/100 dimensions were designed and the behavior of the algorithms was observed. Using F1 unimodal function in Figure 2 (a, b and c) below, F3 basic function in Figure 2 (d, e and f), F6 hybrid function in Figure 2 (g, h and i), F9 composition

function in Figure 2 (j, k and l), box plot graphics of 30, 50 and 100 dimensions were created, respectively.

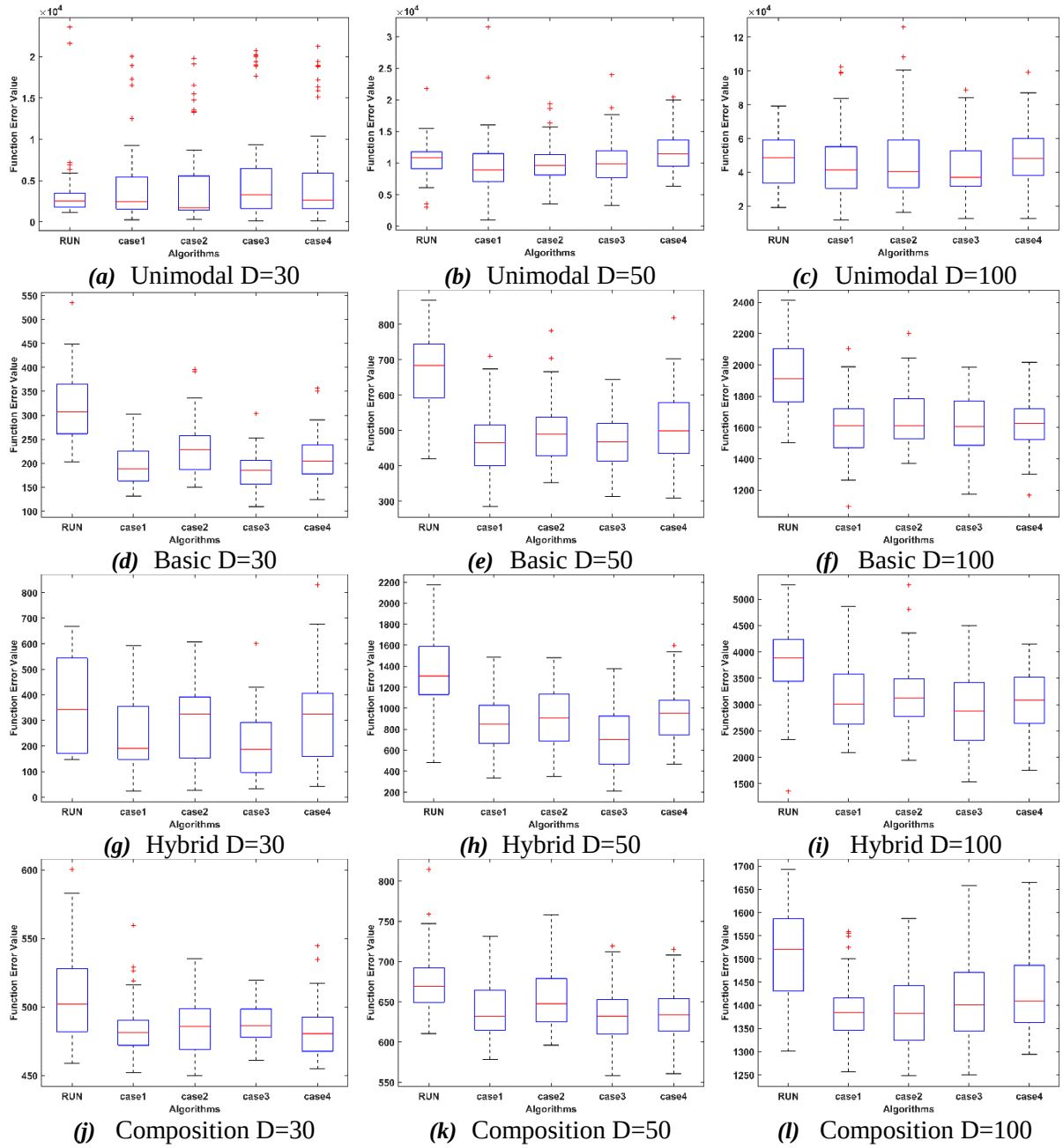

Figure 2. *Box-plots of RUN and FDBRUN algorithms in 30, 50 and 100 dimensions*

As can be seen in Figure 2, box plots of 30, 50 and 100 dimensions are given by using CEC 2020 unimodal, basic, hybrid and composition test functions. Box plots were created by considering the base algorithm (RUN) and the best 4 developed cases. It is seen that all of the developed cases have a lower median value than the base algorithm. Especially when the distribution of the data is examined, the base algorithm shows a wider distribution than the 4 cases compared. The cases tested in the basic, hybrid and composition functions in Figure 2 gave better results in reaching the optimum level compared to the base algorithm. In the unimodal function, it has been observed that the base algorithm performs close to the developed case algorithms.

The first 4 cases clearly showed superiority in terms of performance in large sizes compared to the base algorithm when the box plots are examined as a problem dimensions. Especially in 100 dimensions, this

situation is encountered in all test problems. This situation reveals that the developed cases will offer a more effective solution in high-dimensional problems. In other words, it means that they will converge better to the global minimum or maximum value.

V. CONLUSIONS

In this study, it was aimed to improve the search performance of RUN, a current meta-heuristic search algorithm, in optimization problems. For this purpose, a current selection method, Fitness-Distance Balance (FDB), was used to select candidate solutions in the search process lifecycle. The FDB selection method effectively determined the solution candidates that guided the search process in the RUN algorithm and prevented the algorithm from converging prematurely. Thanks to the FDB method, the RUN algorithm could be designed more compatible with the math-based operation it inspired. Certain experimental studies were conducted to test and verify the performance of the FDB-based RUN algorithm cases developed in the article. In experimental studies, the problems used in CEC 2020 were designed in different dimensions (30/50/100) and the performances of the algorithms in different dimensions were evaluated. Data taken from experimental studies were analyzed using Friedman and Wilcoxon statistical test methods. As a result of these analyzes, it has been clearly demonstrated by the statistical analysis results that the proposed FDBRUN algorithm exhibits a superior performance against the base model (RUN). As the problem dimensions increase, the improved FDBRUN cases yield better results than the base algorithm, which is a clear indicator of the performance of these cases. Thus, it can be said that it will provide more effective results in solving high-dimensional problems. These results show that the FDB method has a significant improvement effect on the search process life cycle in the RUN algorithm. It means that it balances between exploration and exploitation.

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. H. Halim, I. Ismail, and S. Das, "Performance assessment of the metaheuristic optimization algorithms: an exhaustive review," *Artificial Intelligence Revie*w, vol. 54, no. 3, pp. 2323-2409, 2021.

[2] X. S. Yang, "Metaheuristic optimization". *Scholarpedia*, vol. 6, no. 8, 11472, 2011.

[3] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning*," Mach Learn*, vol. 3, pp. 95-99, 1988.

[4] S. Mirjalili, "Genetic algorithm," *In Evolutionary algorithms and neural networks*, Springer, Cham, 2019, pp. 43-55.

[5] D. Bertsimas, and J. Tsitsiklis, "Simulated annealing*," Statistical science*, vol. 8, no. 1, pp. 10-15, 1993.

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

[7] L. Xing, Y. Liu, H. Li, C. C. Wu, W. C. Lin, and X. Chen, "A novel tabu search algorithm for multi-AGV routing problem*," Mathematics*, vol. 8, no. 2, 279, 2020.

[8] K. L. Du and M. N. S. Swamy, “Ant colony optimization,” *In Search and optimization by metaheuristics*. Birkhäuser, Cham, 2016, pp. 191-199.

[9] J. Kennedy and R. Eberhart, “Particle swarm optimization,” *In Proceedings of ICNN'95-international conference on neural networks*, 1995, pp. 1942-1948.

[10] K. L. Du and M. N. S. Swamy, “Particle swarm optimization,” *Search and optimization by metaheuristics*, Birkhäuser, Cham, 2016, pp. 153-173.

[11] 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.

[12] 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.

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

[14] X. S. Yang, and S. Deb, “Cuckoo search via Lévy flights,” *In 2009 World congress on nature & biologically inspired computing*, 2009, pp. 210-214.

[15] R. L. Rardin, and R. Uzsoy “Experimental evaluation of heuristic optimization algorithm: a tutorial”. *J Heuristics*, vol. 7, no. 3, pp. 261–304, 2018.

[16] V. Beiranvand, W. Hare, and Y. Lucet, “Best practices for comparing optimization algorithms,” *Optimization and Engineering*, vol. 18, no. 4, pp. 815-848, 2017.

[17] T. T. Nguyen, S. Yang, and J. Branke, “Evolutionary dynamic optimization: A survey of the state of the art,” *Swarm and Evolutionary Computation*, vol. 6, pp. 1-24, 2012.

[18] J. Tian, C. Sun, Y. Tan, and J. Zeng, “Granularity-based surrogate-assisted particle swarm optimization for high-dimensional expensive optimization,” *Knowledge-Based Systems*, vol. 187, 104815, 2020.

[19] L. Cui, G. Li, Q. Lin, Z. Du, W. Gao, J. Chen, and N. Lu, “A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation,” *Information Sciences*, vol. 367, pp. 1012-1044, 2016.

[20] K. H. Truong, P. Nallagownden, Z. Baharudin, and D. N. Vo, “A quasi-oppositional-chaotic symbiotic organisms search algorithm for global optimization problems*,” Applied Soft Computing*, vol. 77, pp. 567-583, 2019.

[21] A. W. Mohamed and A. K. Mohamed, “Adaptive guided differential evolution algorithm with novel mutation for numerical optimization,” *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 2, pp. 253-277, 2019.

[22] 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.

[23] 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, 105169, 2020.

[24] 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, 107421, 2021.

[25] S. Duman, H. T. Kahraman, U. Guvenc, and S.Aras, "Development of a Lévy flight and FDB-based coyote optimization algorithm for global optimization and real-world ACOPF problems," *Soft Computing*, vol. 25, no. 8, pp. 6577-6617, 2021.

[26] 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, 100821, 2021.

[27] Katı M., Kahraman, H. T. "Arz-Talep tabanlı optimizasyon algoritmasinin FDB yöntemi ile iyileştirilmesi: Mühendislik tasarim problemleri üzerine kapsamli bir araştirma*," Mühendislik Bilimleri ve Tasarım Dergisi*, c. 8, s. 5, ss. 156-172, 2020.

[28] I. Ahmadianfar, A. A. Heidari, A. H. Gandomi, X. Chu, and H. Chen, "RUN beyond the metaphor: an efficient optimization algorithm based on Runge Kutta method," *Expert Systems with Applications*, vol. 181, 115079, 2021.

[29] H. Buch, I. N. Trivedi, and P. Jangir, "Moth flame optimization to solve optimal power flow with non-parametric statistical evaluation validation," *Cogent Engineering*, vol. 4, no. 1, 1286731, 2017.

[30] X. Cai, X. Z. Gao, and Y. Xue, "Improved bat algorithm with optimal forage strategy and random disturbance strategy," *International Journal of Bio-Inspired Computation*, vol. 8, no. 4, pp. 205-214, 2016.

[31] Kahraman, H. T. "Rulet elektromanyetik alan optimizasyon (R-EFO) algoritması," *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, c. 8, s.1, ss. 69-80, 2020.

[32] C. T. Yue, K. V. Price, P. N. Suganthan, J. J. Liang, M. Z. Ali, B. Y. Qu, et al., "Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization," Tech. Rep. Zhengzhou University and Nanyang Technological University, 2019.

[33] F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk, and W. Al-Atabany, "Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems," *Applied Intelligence*, vol. 51, no. 3, pp. 1531-1551, 2021.

[34] Q. Chen, B. Liu, Q. Zhang, J. Liang, P. Suganthan, and B. Qu, "Problem definitions and evaluation criteria for CEC 2015 special session on bound constrained single-objective computationally expensive numerical optimization". Tech. Rep. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Technical Report, Nanyang Technological University, 2014.