Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 1549 – 1566
© TÜBİTAK
doi:10.3906/elk-1908-9

A high-level and adaptive metaheuristic selection algorithm for solving high dimensional bound-constrained continuous optimization problems

Osman GÖKALP[*], **Aybars UĞUR**
Department of Computer Engineering, Faculty of Engineering, Ege University, İzmir, Turkey

Received: 02.08.2019 • **Accepted/Published Online:** 25.11.2019 • **Final Version:** 08.05.2020

Abstract: Metaheuristic algorithms are used to find sufficiently good solutions for the optimization problems that are not solvable in a polynomial time. Although metaheuristics offer a general problem-solving framework and can be applied to various types of optimization problems, their performances depend heavily on the problem to be solved. Thus, hybrid metaheuristics are used to combine strong parts of different algorithms. In this study, a novel adaptive metaheuristic selection algorithm is proposed for solving bound-constrained continuous optimization problems. The developed method hybridizes artificial bee colony, differential evolution, and particle swarm optimization at a high level where each algorithm works independently from each other. As a main contribution to the literature, adaptive selection at metaheuristic level among these three algorithms is achieved by using a rank-based credit assignment and UCB1 multiarmed bandit selection. The effectiveness of the developed algorithm has been evaluated on CEC'17 standard benchmark functions. The obtained numerical results indicate that the proposed algorithm outperforms the individual metaheuristics on which it is built and is more effective especially in high dimensional problems. It is also shown that the proposed algorithm is highly comparable with the related algorithms in the literature. Lastly, a case study that achieves adaptive selection of two good-performing algorithms (namely, covariance matrix adaptation evolution strategy and JADE) for the benchmark used in this study supports the effectiveness of the proposed method.

Key words: Metaheuristics, hybridization, adaptive selection, credit assignment, continuous optimization, multiarmed bandit

1. Introduction

Optimization can simply be defined as choosing the best option from a wide range of alternatives. Most of the optimization problems in daily life are about minimizing the cost or maximizing the profit of a given task. Finding the shortest route in which every city is visited at once [1], optimal placement of wireless sensors for area coverage maximization [2], calculating the optimum structure of pressure vessels or welded beams as engineering design problems [3] are just a few examples of important optimization problems that may be encountered in daily life.

When an optimization problem cannot be solved efficiently by classical algorithms that provide exact solutions, metaheuristic approaches that produce acceptable solutions in a reasonable time are preferred. The main feature of a metaheuristic algorithm is that it provides a general solution framework and can be adapted for different optimization problems. Some of the common metaheuristics that have been successfully applied to solve hard optimization problems are listed as follows: evolutionary strategies [4], genetic algorithms [5],

[*]Correspondence: osman.gokalp@ege.edu.tr

simulated annealing [6], tabu search [7], genetic programming [8], particle swarm optimization [9], and ant colony optimization [10].

The ”no free lunch theorem” of Wolpert and Macready [11] states that no general-purpose optimization algorithm (e.g, metaheuristic) is better than any other, and its performance is equal when all possible problems are taken into account. This means that different optimization algorithms are needed to solve different problems efficiently. Thus, hybrid metaheuristic that combines multiple algorithms can offer a more robust way of dealing with this problem. According to a taxonomy proposed by Talbi [12], hybrid metaheuristics can be divided into two general classes, namely high-level and low-level. In the former, each metaheuristic works itself and is unaware of inner workings of the others. On the other hand, in the latter, a metaheuristic can take part in one of the subfunctions of the others; hence, there is a more interleaved structure.

The main advantage of the high-level hybridization is that member algorithms can be treated as black box, so the hybrid design can easily be extended by including new ones. Moreover, the hybrid design process can be automated by using adaptive selection mechanisms. In this work, we propose a high-level hybrid algorithm which uses adaptive selection of metaheuristics for the purpose of solving bound-constrained continuous optimization problems especially in high dimensions. Specifically, particle swarm optimization (PSO) [9], differential evolution (DE) [13], and artificial bee colony (ABC) [14] metaheuristics are put together and selected adaptively by UCB1 [15] multiarmed bandit algorithm using rank-based credit scores of the algorithms. The reasons we chose these three algorithms are as follows: (i) They are among the most widely used and successful metaheuristics for continuous optimization domain; (ii) they show distinctive search characteristics such as usage of velocity vectors by PSO, two-stage solution update by ABC, and adjustable crossover by DE.

The adaptive algorithm selection for the continuous optimization is not a new concept and has generally been used for solution update strategy decision for a single metaheuristic. For example, SaDE algorithm [16] selects among four mutation formulas of DE, whereas MEABC algorithm [17] selects three mutation formulas of ABC. To the best of our knowledge, there is only one study [18] that selects among different metaheuristics, namely genetic algorithm and DE; however, this selection is made for each solution in the population that metaheuristics operates in a more interleaved way. In this regard, this paper proposes the first high-level hybrid algorithm that makes adaptive selection among standalone metaheuristics for the continuous optimization problems.

In order to test the proposed algorithm, we conducted an experiment on CEC’17 [19] benchmark that includes 30 different numerical optimization functions. The computational results show that high-level adaptive selection-based hybrid algorithm is more successful than individual metaheuristics that make it up and is more effective especially in high dimensional problems. The results also show that the proposed algorithm is highly comparable with the related algorithms in the literature.

The remaining part of the paper is organized as follows. Section 2 gives the mathematical formulation of the bound-constrained continuous optimization algorithms and outlines the three metaheuristics used in this study, namely PSO, DE, and ABC. In Section 3, high-level hybridization is described and the details of adaptive metaheuristic selection mechanism are given. Then, the algorithm of the proposed method is described and detailed in Section 4. After that, the comprehensive computational results of the proposed algorithm is presented and discussed in Section 5. Finally, Section 6 concludes the paper.

2. Metaheuristic algorithms for continuous optimization problems

Let $f : R^D \to R$, where f is the objective function and D is the dimension. The goal of continuous optimization problems is to find $x = (x_1, x_2, \cdots, x_D)$ real number vector that minimizes the value of $f(x)$. If the upper (U) and lower (L) limits for each number in x is given, i.e. $L_i \leq x_i \leq U_i, i = 1, 2, \cdots, D$, the problem is said to be bound-constrained.

2.1. Particle swarm optimization

Particle swarm optimization (PSO) is a swarm intelligence metaheuristic and inspires from the social behavior of bird flocks. A population of PSO consist of particles, which correspond to candidate solutions, and defined by two real number vectors, namely position and velocity. The position vector of a particle, $\vec{X} = (x_1, x_2, \cdots, x_D)$, is the actual solution, whereas the velocity vector, or $\vec{V} = (v_1, v_2, \cdots, v_D)$, gives the direction that the position is moved, where D is the problem dimension.

At each iteration of PSO, the following updates are made for each particle in the population $P = \{(\vec{X}_i, \vec{V}_i) : 1 \leq i \leq NP\}$, where NP is the population size. Firstly, the velocity of a particle is adjusted using both its personal best position, or $\vec{P}$ and the global best position of the entire population, or $\vec{G}$, as in (1). The parameters $w, c1$, and $c2$ are used for controlling the behavior of the optimization process, whereas $r1$ and $r2$ are the random numbers between (0,1). Then, the calculated velocity is added to the previous position of the particle and particle is moved to new position as in (2). After that, both the particle's personal best and the global best positions are updated if there is an improvement in the objective value (see (3) and (4)). Finally, when the maximum number of iterations is reached, the best solution found is returned by the algorithm.

$$\vec{V}_i = w\vec{V}_i + c_1 r_1(\vec{P}_i - \vec{X}_i) + c_2 r_2(\vec{G} - \vec{X}_i), \tag{1}$$

$$\vec{X}_i = \vec{X}_i + \vec{V}_i, \tag{2}$$

$$\vec{P}_i = \begin{cases} \vec{X}_i, & \text{if } f(\vec{X}_i) \leq f(\vec{P}_i) \\ \vec{P}_i, & \text{otherwise} \end{cases}, \tag{3}$$

$$\vec{G} = \begin{cases} \vec{P}_i, & \text{if } f(\vec{P}_i) \leq f(\vec{G}) \\ \vec{G}, & \text{otherwise} \end{cases}. \tag{4}$$

2.2. Differential evolution

Differential evolution (DE) is a population-based evolutionary metaheuristic algorithm. Each candidate solution in DE is represented by a real number vector $\vec{X} = (x_1, x_2, \cdots, x_D)$, where D is the problem dimension. A population of candidate solutions $P = \{X_i : 1 \leq i \leq NP\}$, where NP is the population size, is updated through iterations to find the global optimum solution.

At each iteration of DE, the following updates are made for each individual in the population. Firstly, the target vector $\vec{V}$ is generated using (5), where $r1$, $r2$, and $r3$ are randomly picked indices in $[1, NP]$ and satisfies the condition $r1 \neq r2 \neq r3 \neq i$. Then, the trial vector $\vec{U}$ is generated using (6), where CR is the crossover rate and I is the random dimension index which is used to ensure that at least one dimension update is made. Lastly, the trial vector replaces the current individual using (7). When the maximum number of iterations is reached, the best solution found is returned by the algorithm.

$$\vec{V}_i = \vec{X}_{r1} + F(\vec{X}_{r2} - \vec{X}_{r3}) \tag{5}$$

$$\vec{U}_{i,j} = \begin{cases} \vec{V}_{i,j}, & \text{if } rand(0,1) \le \text{CR or } j = I \\ \vec{X}_{i,j}, & \text{otherwise} \end{cases} \tag{6}$$

$$\vec{X}_i = \begin{cases} \vec{U}_i, & \text{if } f(\vec{U}_i) \le f(\vec{X}_i) \\ \vec{X}_i, & \text{otherwise} \end{cases} \tag{7}$$

2.3. Artificial bee colony

Artificial bee colony (ABC) is a swarm intelligence metaheuristic and is inspired by the foraging behavior of honey bees. The ABC algorithm consists of four main components, namely food sources, employed bees, onlooker bees, and scout bees. Food sources correspond to candidate solutions of an optimization problem that is to be solved. On the other hand, the three types of bees represent three different search strategies. Specifically, each employed bee is associated with one food source and search its neighborhood to find a better one. Onlooker bees work similar with the employed bees; however, they are not associated with the current food sources directly. Rather, they probabilistically choose better food sources. In this way, some of the food sources (probably the worse ones) may not be visited by onlooker bees. This behavior allows to exploit promising solutions several times. Therefore, it strengthens the exploitation behavior of the search process. As a last group of bees, the duty of the scout bees is to find completely new food sources when the current ones all exhausted.

In ABC, each solution is represented by a real number vector $\vec{X} = (x_1, x_2, \cdots, x_D)$, where D is the dimension of the problem. At each iteration of ABC, three search phases are applied to the population of solutions $P = \{\vec{X}_i : 1 \le i \le NP\}$, where NP is the population size. In the first phase, which is called employed bees, the new neighbor food source $\vec{V}$ is searched for each solution i using (8), $k \ne i$ is the random picked solution index in P, j is the randomly picked dimension index, and ϕ is the random real number in the range $(-1, 1)$. Then the greedy selection is applied to neighbor food source using (9), where $f(.)$ is the objective function. In the second phase, which is called onlooker bees, (8) and (9) are again used to find and accept new food sources, respectively; however, this time only the promising food sources are considered. That is, the probabilistic selection is made using (10) and (11), where function $fit(.)$ calculates the fitness value of a given solution. The last phase is called scout bees that replaces one of the food sources with a new random one if it is exhausted, i.e. it does not give a lead to find better neighbor food source for a predefined limit value. After reaching a maximum number of iterations, ABC algorithm terminates and returns the best solution found so far.

$$\vec{V}_{i,j} = \vec{X}_{i,j} + \phi_i(\vec{X}_{i,j} - \vec{X}_{k,j}), \tag{8}$$

$$\vec{X}_i = \begin{cases} \vec{V}_i, & \text{if } f(\vec{V}_i) \leq f(\vec{X}_i) \\ \vec{X}_i, & \text{otherwise} \end{cases}, \tag{9}$$

$$fit(\vec{X}_i) = \begin{cases} \frac{1}{1+f(\vec{X}_i)}, & \text{if } f(\vec{X}_i) \geq 0 \\ 1 + \mid f(\vec{X}_i) \mid, & \text{otherwise} \end{cases}, \tag{10}$$

$$p_i = \frac{fit(\vec{X}_i)}{\sum_{j=1}^{NP} fit(\vec{X}_j)}. \tag{11}$$

3. A high-level and adaptive metaheuristic selection

For high-level hybrid structures, it is necessary to decide which metaheuristic is to be operated within a certain period of time. In case of sequential hybrids, i.e. concurrent running of the metaheuristics is not the case, two approaches can be employed for the selection of metaheuristics: (i) the use of a predefined sequence and (ii) the use of a real-time and adaptive mechanism. Figures 1a and 1b show flowchart diagrams showing the approaches (i) and (ii), respectively, for 3 metaheuristics. The main drawback of the former approach is to determine which sequence to use before the algorithm starts. Thus, this affects the usability of the algorithm negatively. Another drawback of this method is that optimization itself is a dynamic process. That is, one metaheuristic which is good at the beginning may not necessarily perform well at the later stages. Fortunately, the latter approach can provide a solution to these problems by allowing adaptive selection of metaheuristics while the algorithm is running. The adaptive selection method used in this study consists of two parts that are credit assignment and selection rule and they are explained in the following subsections, respectively.

3.1. Credit assignment

In order to decide which metaheuristic to be selected for a given time period, their performance should be monitored continuously and a credit value that shows how much the algorithm is preferable should be assigned to each one. In our case, the credit should depend on how much the objective function value of the continuous optimization problem is improved. However, two problems may be encountered at this point. Firstly, the performance of the algorithms used in solving optimization problems may vary in different times or stages of the optimization process. Secondly, using the objective value improvement directly can raise the problem of scaling because of the fact that the amount of change in the objective function tends to get smaller as the algorithm converges.

In this study, we used the sum of ranks (SR) method [20] to calculate credits of metaheuristics because it addresses both problems stated above. To be more specific, SR uses sliding window (SW) data structure that stores only the recent data about algorithm performances. Thus, the credit values are kept up-to-date for the current time interval. Moreover, SR uses ranked values instead of raw objective value improvements, so the scaling problem is solved. Equation 12 formulates the calculation of SR-based credit for the algorithm i at time t, where $D \in [0, 1]$ is the decay factor and W is the sliding window size. As D gets close to 0, the effect of the higher ranked algorithms increases. On the other hand, as D gets close to 1, the effect of the ranks is

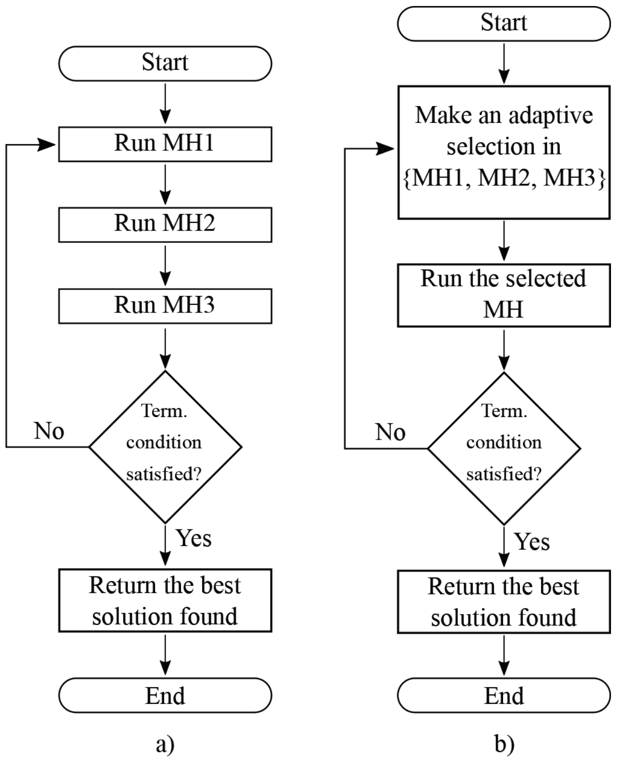
Start
Run MH1
Run MH2
Run MH3
Term. condition satisfied?
No
Yes
Return the best solution found
End
a)
Start
Make an adaptive selection in {MH1, MH2, MH3}
Run the selected MH
Term. condition satisfied?
No
Yes
Return the best solution found
End
b)

Figure 1. A flowchart diagram for running 3 metaheuristics (MH1, MH2, MH3) with high-level and sequential hybrid architecture a) using predefined order b) using adaptive selection.

decreased.

$$SR_{i,t} = \frac{\sum_{alg_r=i} D^r(W-r)}{\sum_{r=1}^{W} D^r(W-r)} \tag{12}$$

Figure 2 demonstrates a simple example of SW for 3 algorithms. The size of the SW is W=4, meaning that only the last 4 results for the application of the algorithms are taken into account. Ranks (r) are given to each algorithm according to Δ values that correspond to objective value improvements. Using Figure 2, an example credit calculation for algorithm 0 at time 8 with $D = 0.5$ can be done by (12) as follows:

$$SR_{0,8} = \frac{0.5^0(4-0) + 0.5^2(4-2)}{0.5^0(4-0) + 0.5^1(4-1) + 0.5^2(4-2) + 0.5^3(4-3)} = 0.48$$

3.2. Selection rule

Selecting which algorithm to run from a set of alternatives is an example of exploration/exploitation dilemma. In other words, choosing the best-performing algorithm so far (pure exploitation) may cause the selection process to miss better possible options. On the other hand, only random selection of algorithms (pure exploration) means that no previously available useful information is used. Because UCB1 [15], one of the multiarmed bandit problem algorithms, provides a simple yet efficient solution to this dilemma, we used it as a selection rule for the metaheuristics in this paper.

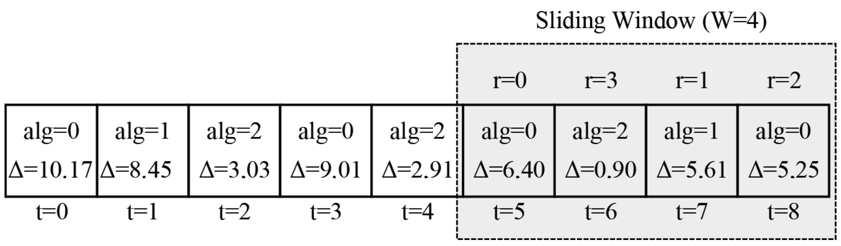

Figure 2. An example of sliding window data structure for 3 algorithms for time t=5 to t=8.

The selection formula of UCB1 is given in (13), where $\hat{x}$ is the average reward (current credit value) of algorithm j, n_j is the number of times j has been selected so far, and n is the total number of selections for all the algorithms. Because the first term of the formula uses a credit value, it supports exploitation. On the other hand, the second term punishes the frequently selected algorithms; therefore, it supports exploration. At this point, parameter C is used for controlling the balance between these behaviors, i.e. the higher C is, the more the exploration is, and vice versa.

$$\operatorname*{argmax}_j(\hat{x}_j + C\sqrt{\frac{2\log n}{n_j}}) \tag{13}$$

4. The proposed algorithm

The proposed algorithm is based on high-level hybridization of three main metaheuristics, namely PSO, DE, and ABC, for solving the bound-constrained continuous optimization problems. Specifically, three metaheuristics are adaptively selected and run in a sequential way. The selection process uses success rate-based credit assignment and UCB1 selection methods, which were explained previously in Section 3.1 and Section 3.2, respectively.

For the implementations of PSO, DE, and ABC, we used the most standard versions in the literature as listed below:

- PSO: SPSO-2011 algorithm of Zambrano-Bigiarini et al. [21].
- DE: DE algorithm with “DE/rand/1” selection strategy of Storn and Price [13].
- ABC: Standard ABC algorithm of Karaboga [22].

The flowchart of the algorithm of the proposed method is given in Figure 3. In the beginning, the PSO, ABC, and DE metaheuristics are initialized with predefined parameter values. Then, the population, which includes a pool of candidate solutions for a given optimization problem, is created randomly. After that, the main loop of the algorithm starts and continues until the function evaluations (FEs) count reaches the budget (MAX_FEs).

At each iteration of the main loop, firstly, up-to-date credits are calculated using sliding window (SW) and assigned to each metaheuristic. Secondly, using the credit values, one of the metaheuristics is selected by the UCB1 algorithm. Then, the selected metaheuristic is run for the L FEs and the improvement obtained on the best solution of population is stored in SW. After the current FE count is updated, L is also updated by multiplying with a small constant to run selected metaheuristic with larger FE budgets as selection algorithm itself converges. After some preliminary testing, we set this small multiplier constant to 1.05. Finally, at the last step of the iteration, the current FEs is checked and the algorithm decides either to continue to the next iteration or to terminate and return the best solution found.

It should be noted that the same population is optimized by each of selected metaheuristic. However, contrary to other algorithms, PSO uses velocity vectors in addition to the position vectors, or solutions. Therefore, when the selected metaheuristic is PSO, its velocity vectors are re-initialized while the position vectors are taken directly from the population.

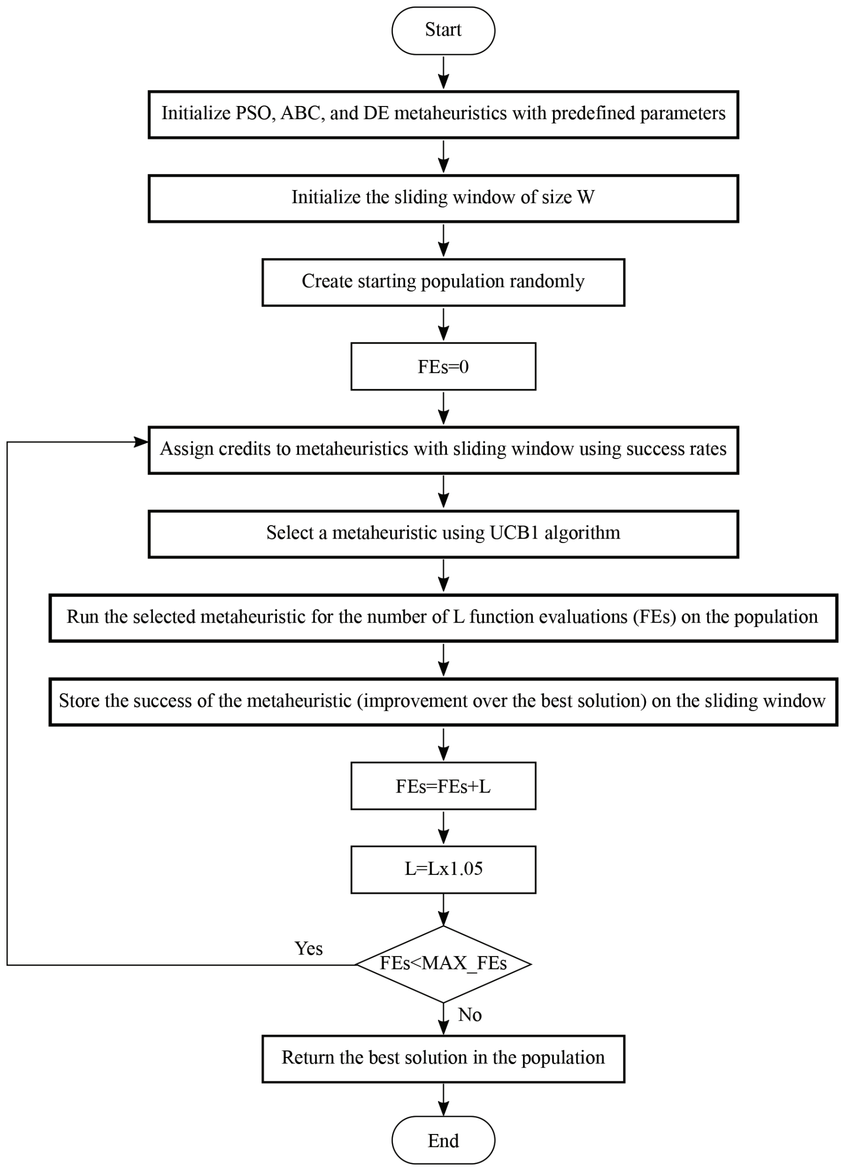
Start
Initialize PSO, ABC, and DE metaheuristics with predefined parameters
Initialize the sliding window of size W
Create starting population randomly
FEs=0
Assign credits to metaheuristics with sliding window using success rates
Select a metaheuristic using UCB1 algorithm
Run the selected metaheuristic for the number of L function evaluations (FEs) on the population
Store the success of the metaheuristic (improvement over the best solution) on the sliding window
FEs=FEs+L
L=Lx1.05
Yes
FEs<MAX_FEs
No
Return the best solution in the population
End

Figure 3. A flowchart diagram of the proposed algorithm.

5. Experimental work

5.1. Benchmark functions

In this study, we used CEC'17 benchmark that includes 30 functions in 4 categories for single objective bound-constrained continuous optimization problems. These functions are shifted and rotated to reduce the bias of the developed optimization algorithms. In addition, functions are scalable to 4 different problem dimensions that are 10, 30, 50, and 100. The functions, their categories, and problem searches, and optimum objective (f^*) values are listed in Table 1. Hybrid functions combine N basic functions as subcomponents, whereas composition functions combine N basic functions with weight and bias. The further details about the benchmark and definitions of basic functions can be found in [19].

5.2. Experimental setup

For the experimental setup of this study, we followed the rules below:

- Each algorithm is run 51 times with different random seeds. To make fair comparison between algorithms, the starting populations are kept the same for each run number, i.e. 51 different initial populations are generated and used by all the algorithms.
- Initial population generation is done randomly using uniform distribution in range $[-100, 100^D]$, where D is the problem dimension.
- The range for searching over the test functions are also determined as $[-100, 100^D]$.
- The budget of optimization, or maximum number of function evaluations, is $10,000 \times D$ for each algorithm.
- The quality of each solution is assessed by its error value that is calculated by $f(X) - f^*$, where f^* is the best objective function value that can be obtained.
- If the error is equal or smaller than the threshold value of 10^{-8}, it is accepted as 0.

5.2.1. Parameter setting

In order the set of the 4 main parameters of the proposed algorithm that are L: FE count for each metaheuristic run, W: the sliding window size, D: decay factor of the UCB1 selection, and C: balancing factor of the UCB1 selection, we used the irace package [23] which is an automatic algorithm configuration tool based on iterated racing procedure.

In the configuration of irace, maximum budget was set to 5000. Taking dimensions of 10, 30, and 50, odd-numbered functions were used as training set and even-numbered functions were used as test set. Other configurations were kept as default and the tuned parameter values obtained are shown in Table 2.

Moreover, for the inner parameters of DE and ABC, we used the default values in [24] and for the PSO, we used the default values in [21].

5.3. Computational results and comparisons

5.3.1. Comparison of the proposed algorithm with its constituent metaheuristics

To evaluate the performance of the proposed algorithm, we first make a comparison with its constituent metaheuristics in Tables 3 and 4 for different problem dimensions. The first statistical assessment is performed at

Table 1. CEC'17 test functions and their properties.

Func. #	Category	Problem search	f^*
1	Unimodal	Shifted and rotated bent cigar function	100
2		Shifted and rotated sum of different power function	200
3		Shifted and rotated zakharov function	300
4	Simple multimodal	Shifted and rotated Rosenbrock's function	400
5		Shifted and rotated Rastrigin's function	500
6		Shifted and rotated expanded Scaffer's F6 function	600
7		Shifted and rotated Lunacek Bi_Rastrigin function	700
8		Shifted and rotated noncontinuous Rastrigin's function	800
9		Shifted and rotated Levy function	900
10		Shifted and rotated Schwefel's function	1000
11	Hybrid	Hybrid function 1 (N=3)	1100
12		Hybrid function 2 (N=3)	1200
13		Hybrid function 3 (N=3)	1300
14		Hybrid function 4 (N=4)	1400
15		Hybrid function 5 (N=4)	1500
16		Hybrid function 6 (N=4)	1600
17		Hybrid function 6 (N=5)	1700
18		Hybrid function 6 (N=5)	1800
19		Hybrid function 6 (N=5)	1900
20		Hybrid function 6 (N=6)	2000
21	Composition	Composition function 1 (N=3)	2100
22		Composition function 2 (N=3)	2200
23		Composition function 3 (N=4)	2300
24		Composition function 4 (N=4)	2400
25		Composition function 5 (N=5)	2500
26		Composition function 6 (N=5)	2600
27		Composition function 7 (N=6)	2700
28		Composition function 8 (N=6)	2800
29		Composition function 9 (N=3)	2900
30		Composition function 10 (N=3)	3000
Bound constraint: $[-100, 100]^D$			

Table 2. The parameters of the proposed algorithm and the tuned values by irace.

Param. name	Param. type	Param. range	irace value
L	integer	[250,1500]	608
W	integer	[5,40]	31
D	real	[0.1,1.0]	0.32
C	real	[0.01,5.0]	2.87

function level using the Mann–Whitney U-test (also known as the Wilcoxon rank-sum), which is a nonparametric method to check whether two independent samples are significantly different from each other. The results for this test are presented in column "T". According to this, "+", "-", and "=" values indicate that the proposed algorithm is better than, worse than, and equal to the compared algorithm for a given function with respect to $P = 0.05$ significance level. The overall results presented in row "+/=/-" indicate that the proposed algorithm outperforms all its subalgorithms except DE in 10 dimensional problems.

In addition, another evaluation between the algorithms is performed by calculating the Friedman ranking scores. The results show that the proposed algorithm reaches the better ranking for all dimensions. Moreover, the statistical assessment is performed at benchmark level using the Wilcoxon signed-rank (W.S.R.) test, which is a nonparametric method to check whether two related or paired samples are significantly different from each other. The P-values below 0.05 for W.S.R. prove that the difference between the proposed algorithm and the compared algorithms is significant when the whole benchmark is considered. The results show that the proposed algorithm outperforms all of its constituent metaheuristics in higher problem dimensions such as 50 and 100.

Finally, Figure 4 shows the convergence behaviors of the algorithms with problem dimension of 50 for 4 selected functions from each category. These functions are listed as: F3 from unimodal functions, F10 from simple multimodal functions, F17 from hybrid functions, and F24 from composition functions. It is seen that the convergence behaviors of the algorithms can vary according to the function to be solved. For instance, ABC shows premature convergence and differs from the others for F3, whereas DE shows relatively slow convergence in comparison with other 3 algorithms for F10. It can also be seen that all the algorithms show similar behavior for F17. Lastly, for F24, the proposed algorithm converges faster than the others.

5.3.2. Comparison of the proposed algorithm with other algorithms in the literature

This section compares the performance of the proposed algorithm with some of the related adaptive algorithms in the literature. For this purpose we selected the following algorithms: (i) MEABC [17] that adaptively selects among three ABC solution update strategies, (ii) SaDE [16] that adaptively selects among four DE solution update strategies, (iii) PPSO [25] that adapts parameters at particle level, and (iv) CMA-ES [26], which is based on covariance matrix adaptation for evolution strategies and one of the most successful algorithms in continuous optimization field.

For all the compared algorithms the same population size (50) with the proposed algorithm were used. The computational results for the problem dimension of 50 are reported in Table 5. According to Friedman scores our algorithm takes the second place after the CMA-ES. Similarly, when function based-comparisons (+/=/-) are evaluated, overall results suggest that the proposed algorithm outperforms others except CMA-ES. Finally, the algorithm level evaluations using W.S.R. test indicates that the difference between our algorithm and MEABC, SaDE, and PPSO algorithms is significant when the whole benchmark is considered with respect to P values smaller than 0.05. Therefore, we can conclude that the proposed algorithm is comparable with the other algorithms in the literature.

Table 3. Comparison of the proposed algorithm with ABC, DE, and PSO for D=10 and D=30.

Func.	D=10							D=30						
	ABC		DE		PSO		Proposed	ABC		DE		PSO		Proposed
	Mean Err.	T.	Mean Err.	T.	Mean Err.	T.	Mean Err.	Mean Err.	T.	Mean Err.	T.	Mean Err.	T.	Mean Err.
1	2.27E+02	+	0.00E+00	-	8.85E+02	+	6.67E+00	1.88E+02	-	3.13E+03	=	2.69E+03	=	2.81E+03
2	4.29E-04	+	0.00E+00	=	2.27E-04	+	4.90E-08	6.74E+09	+	1.27E+17	+	1.43E+07	+	1.17E+08
3	6.26E+03	+	0.00E+00	=	0.00E+00	=	2.85E-05	1.24E+05	+	1.60E-05	-	1.06E+04	+	1.24E+02
4	1.47E-01	-	4.44E-01	-	2.41E+00	+	1.01E+00	2.79E+01	-	5.37E+01	-	6.67E+01	=	7.26E+01
5	7.71E+00	+	7.07E+00	+	7.59E+00	+	1.62E+00	8.28E+01	+	3.75E+01	+	7.96E+01	+	1.22E+01
6	0.00E+00	-	2.79E-08	-	9.21E-01	+	1.60E-05	0.00E+00	-	5.79E-04	+	2.38E+01	+	1.32E-03
7	1.77E+01	+	1.98E+01	+	1.50E+01	+	1.26E+01	9.96E+01	+	1.26E+02	+	1.47E+02	+	4.36E+01
8	9.21E+00	+	5.95E+00	+	5.97E+00	+	1.77E+00	9.48E+01	+	2.82E+01	+	6.39E+01	+	1.27E+01
9	4.30E-02	+	0.00E+00	=	2.71E+00	+	0.00E+00	8.30E+02	+	2.95E-01	-	1.03E+03	+	2.27E+00
10	2.58E+02	+	3.29E+02	+	6.19E+02	+	7.99E+00	2.31E+03	+	4.21E+03	+	3.12E+03	+	1.29E+03
11	4.46E+00	+	7.96E-01	-	2.52E+01	+	1.36E+00	5.50E+02	+	1.81E+01	+	1.09E+02	+	1.48E+01
12	2.70E+04	+	6.75E+01	=	9.27E+03	+	1.20E+02	4.12E+05	+	1.59E+04	=	5.08E+04	+	1.67E+04
13	3.11E+02	+	4.41E+00	-	6.46E+03	+	5.39E+00	6.59E+03	+	1.23E+03	-	9.93E+03	+	6.17E+03
14	2.72E+02	+	5.46E-01	-	8.02E+02	+	8.26E-01	6.81E+04	+	2.72E+01	+	7.19E+03	+	2.51E+01
15	1.49E+02	+	2.87E-01	=	2.30E+03	+	3.25E-01	1.56E+03	+	1.08E+01	-	1.90E+03	+	1.74E+01
16	5.05E+00	+	5.87E+00	=	1.69E+02	+	9.43E-01	6.22E+02	+	4.79E+02	+	9.07E+02	+	7.93E+01
17	2.32E+00	+	5.37E+00	=	4.80E+01	+	1.51E+00	2.03E+02	+	7.61E+01	+	3.74E+02	+	4.25E+01
18	1.15E+03	+	2.94E+00	=	5.78E+03	+	2.66E+01	1.68E+05	+	1.52E+03	=	7.91E+04	+	3.45E+03
19	8.01E+01	+	1.25E-01	=	4.22E+03	+	4.43E-02	1.38E+03	+	8.97E+00	-	3.53E+03	+	2.21E+01
20	5.57E-01	+	9.35E-01	=	5.18E+01	+	6.67E-01	2.43E+02	+	8.69E+01	=	3.17E+02	+	9.34E+01
21	1.13E+02	-	2.03E+02	+	2.08E+02	+	1.89E+02	2.63E+02	+	2.34E+02	+	2.58E+02	+	2.14E+02
22	6.19E+01	-	9.84E+01	=	1.00E+02	=	1.00E+02	6.30E+02	+	2.39E+03	+	1.05E+02	+	1.00E+02
23	3.08E+02	+	3.07E+02	+	3.11E+02	+	3.02E+02	4.15E+02	+	3.79E+02	+	4.66E+02	+	3.57E+02
24	1.01E+02	-	3.32E+02	+	3.31E+02	+	3.26E+02	4.29E+02	=	4.56E+02	+	5.36E+02	+	4.32E+02
25	1.84E+02	-	4.11E+02	=	4.31E+02	+	4.15E+02	3.82E+02	-	3.87E+02	+	4.12E+02	+	3.87E+02
26	9.55E+01	-	3.29E+02	-	3.31E+02	=	2.98E+02	2.96E+02	-	1.24E+03	+	2.31E+03	+	1.03E+03
27	3.95E+02	+	3.90E+02	+	3.99E+02	+	3.89E+02	5.14E+02	+	5.07E+02	=	5.72E+02	+	5.05E+02
28	2.60E+02	-	4.02E+02	-	5.52E+02	+	4.63E+02	4.01E+02	+	3.71E+02	=	3.74E+02	+	3.38E+02
29	2.53E+02	+	2.31E+02	-	2.93E+02	+	2.42E+02	6.36E+02	+	4.60E+02	=	1.08E+03	+	4.35E+02
30	1.96E+04	+	8.19E+04	=	1.83E+05	+	4.64E+04	6.01E+03	+	2.27E+03	-	5.11E+03	+	3.22E+03
+/=/-	**22/0/8**		8/13/9		**27/0/3**		-	**24/1/5**		**16/7/7**		**28/2/0**		-
Fried.	2.40		2.03		3.67		**1.90**	2.87		2.08		3.37		**1.68**
W.S.R.	P=0.111		P=0.538		**P=0.000**		-	**P=0.000**		P=0.163		**P=0.000**		-

Table 4. Comparison of the proposed algorithm with ABC, DE, and PSO for D=50 and D=100.

Func.	D=50							D=100						
	ABC		DE		PSO		Proposed	ABC		DE		PSO		Proposed
	Mean Err.	T.	Mean Err.	T.	Mean Err.	T.	Mean Err.	Mean Err.	T.	Mean Err.	T.	Mean Err.	T.	Mean Err.
1	1.69E+03	-	5.84E+03	=	2.62E+03	=	4.18E+03	2.33E+03	-	8.20E+03	=	7.31E+03	=	6.74E+03
2	4.11E+18	+	1.49E+38	+	2.44E+21	+	1.81E+08	3.16E+62	+	4.88E+93	+	2.82E+79	+	2.54E+44
3	2.68E+05	+	5.63E+03	=	5.30E+04	+	9.44E+03	6.52E+05	+	2.11E+05	+	1.96E+05	+	1.39E+05
4	4.17E+01	=	8.28E+01	=	1.11E+02	+	6.22E+01	2.04E+02	=	2.02E+02	=	2.78E+02	+	2.10E+02
5	2.00E+02	+	6.84E+01	+	1.71E+02	+	3.34E+01	6.75E+02	+	1.64E+02	+	5.28E+02	+	1.26E+02
6	0.00E+00	-	1.06E-02	+	3.84E+01	+	4.26E-03	0.00E+00	-	9.09E-01	=	4.76E+01	+	1.75E+00
7	2.16E+02	+	2.37E+02	+	4.06E+02	+	8.48E+01	6.79E+02	+	4.53E+02	+	1.59E+03	+	2.66E+02
8	2.01E+02	+	6.86E+01	+	1.78E+02	+	3.15E+01	7.06E+02	+	1.68E+02	+	5.82E+02	+	1.25E+02
9	4.82E+03	+	1.61E+01	=	5.49E+03	+	2.39E+01	3.06E+04	+	6.51E+02	+	2.01E+04	+	4.35E+02
10	4.21E+03	+	1.22E+04	+	5.94E+03	+	2.61E+03	1.11E+04	+	2.99E+04	+	1.34E+04	+	8.14E+03
11	1.65E+03	+	5.20E+01	=	1.76E+02	+	6.08E+01	6.60E+04	+	1.75E+02	-	1.17E+03	+	4.33E+02
12	2.72E+06	+	8.24E+04	-	7.78E+05	+	1.38E+05	1.44E+07	+	5.77E+05	=	2.06E+06	+	6.07E+05
13	3.55E+03	=	8.25E+03	+	6.17E+03	+	3.92E+03	2.39E+03	=	6.40E+03	=	8.50E+03	+	3.71E+03
14	8.63E+05	+	6.40E+02	=	2.94E+04	+	5.96E+02	3.80E+06	+	2.08E+04	-	1.11E+05	+	3.18E+04
15	8.72E+03	+	3.36E+03	=	4.81E+03	=	4.06E+03	9.92E+02	=	4.13E+03	=	1.59E+03	=	2.06E+03
16	1.28E+03	+	1.18E+03	+	1.50E+03	+	3.91E+02	3.26E+03	+	3.34E+03	+	4.00E+03	+	1.65E+03
17	9.62E+02	+	8.99E+02	+	1.45E+03	+	3.34E+02	2.66E+03	+	2.70E+03	+	3.35E+03	+	1.39E+03
18	8.20E+05	+	9.43E+03	=	9.86E+04	+	1.06E+04	3.41E+06	+	1.49E+05	=	3.62E+05	+	1.49E+05
19	7.27E+03	+	1.25E+03	-	1.33E+04	+	6.63E+03	1.34E+03	=	6.52E+03	+	2.33E+03	+	1.95E+03
20	7.63E+02	+	8.54E+02	+	7.75E+02	+	1.67E+02	2.54E+03	+	3.15E+03	+	2.52E+03	+	9.38E+02
21	4.07E+02	+	2.88E+02	+	3.56E+02	+	2.34E+02	9.04E+02	+	3.92E+02	+	8.32E+02	+	3.56E+02
22	4.94E+03	+	1.25E+04	+	6.49E+03	+	4.49E+02	1.27E+04	+	3.03E+04	+	1.62E+04	+	6.12E+03
23	6.48E+02	+	4.79E+02	+	7.62E+02	+	4.50E+02	8.83E+02	+	7.21E+02	+	1.49E+03	+	6.39E+02
24	1.03E+03	+	5.88E+02	+	8.16E+02	+	5.32E+02	1.48E+03	+	1.08E+03	+	2.24E+03	+	9.95E+02
25	5.10E+02	=	5.26E+02	=	5.59E+02	+	5.15E+02	7.19E+02	-	7.74E+02	=	8.23E+02	+	7.67E+02
26	1.63E+03	=	1.71E+03	+	5.57E+03	+	1.49E+03	9.12E+03	+	5.24E+03	+	1.98E+04	+	4.39E+03
27	6.47E+02	+	6.04E+02	+	1.01E+03	+	5.42E+02	7.59E+02	+	7.06E+02	+	1.60E+03	+	6.74E+02
28	4.83E+02	=	4.93E+02	=	5.01E+02	+	4.80E+02	5.80E+02	+	5.62E+02	=	6.08E+02	+	5.71E+02
29	1.11E+03	+	4.98E+02	=	2.01E+03	+	4.77E+02	4.05E+03	+	1.69E+03	-	4.76E+03	+	2.11E+03
30	7.02E+05	+	6.50E+05	=	8.96E+05	+	6.38E+05	1.22E+04	+	6.54E+03	=	1.13E+04	+	6.61E+03
+/=/-	23/5/2		16/12/2		28/2/0		-	23/4/3		17/10/3		28/2/0		-
Fried.	2.77		2.37		3.43		1.43	2.76		2.33		3.34		1.57
W.S.R.	P=0.000		P=0.023		P=0.000		-	P=0.000		P=0.016		P=0.000		-

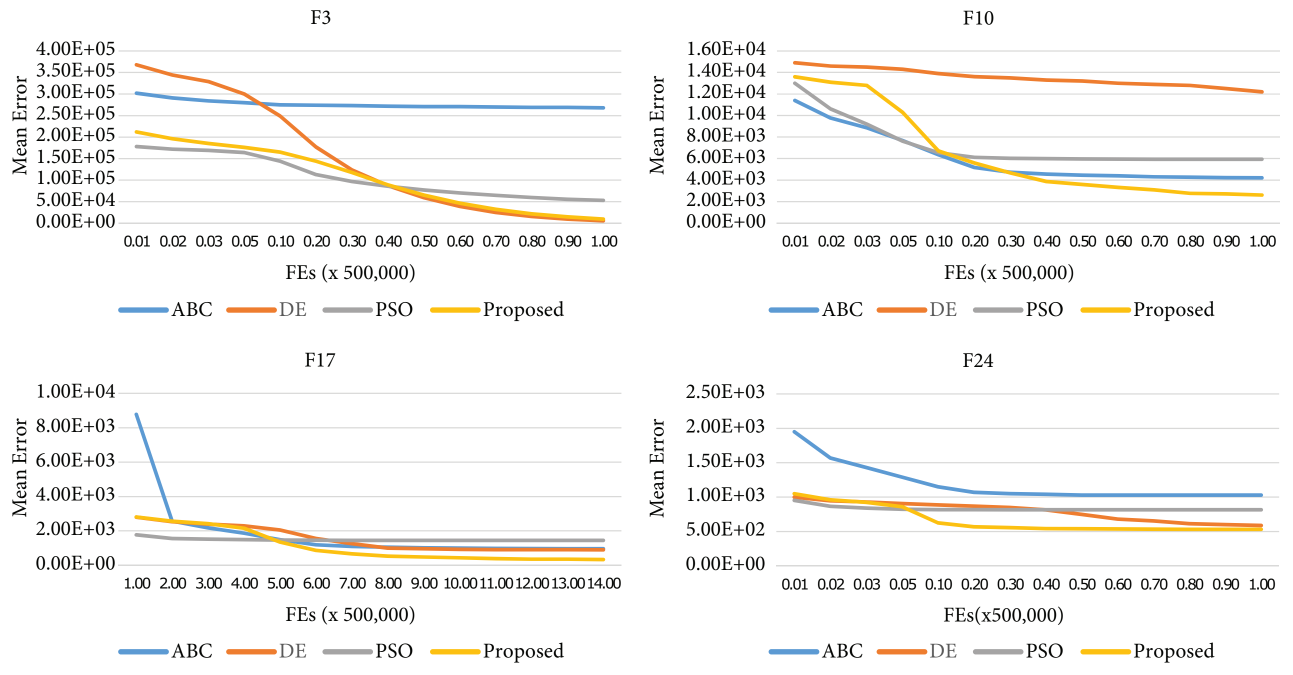

Figure 4. Convergence graphs for the proposed algorithm and its constituent algorithms for functions 3, 10, 17, 24 with D=50.

5.4. A case study: adaptive metaheuristic selection with two good-performing algorithms

After validating the performance of the proposed algorithm with classical numerical optimization algorithms such as ABC, DE, and PSO, in this section we also test the adaptive selection algorithm using two new metaheuristics: JADE [27] and covariance matrix adaptation evolution strategy (CMA-ES), the variants of which have shown high performance to solve the CEC benchmark functions before in [28] and [29]. Note that, because CMA-ES uses a single initial solution to start the search, we used the population's best solution when this algorithm is selected and will be executed.

For the experimental design, we used the same setting that is defined in Section 5.2. After tuning the parameters with irace as it is explained in Section 5.2.1, we obtained the following values to be used in the computational experiment: L=1124, W=17, D=0.39, and C=4.90.

The computational results for adaptive metaheuristic selection of JADE and CMA-ES algorithms with the proposed method in this study are presented in Table 6. The comparison between the adaptive selection and the constituent algorithms are made at function level using the Mann–Whitney U-test based on mean errors, which are presented in column "T.". That is, "+", "-", and "=" values indicate that the adaptive metaheuristic selection is better than, worse than, and equal to the compared algorithm for a given function with respect to $P = 0.05$ significance level.

The overall results suggest that the adaptive selection could improve results of 22 functions for JADE and 19 functions for CMAES. It can also be seen that the improvements are mainly obtained for simple multimodal (F4-F10) and composition function (F21-F30) groups.

Table 5. Comparison of the proposed algorithm with other algorithms in the literature for D=50 based on mean errors.

Func.	MEABC	SaDE	PPSO	CMA-ES	Proposed
1	6.60E+03	4.91E+02	3.89E+02	0.00E+00	4.18E+03
2	8.29E+23	8.19E+18	1.25E+12	0.00E+00	1.81E+08
3	2.55E+05	6.43E+04	8.65E+02	0.00E+00	9.44E+03
4	5.60E+01	4.55E+01	9.13E+01	4.34E+01	6.22E+01
5	1.25E+02	6.89E+01	2.01E+02	3.48E+01	3.34E+01
6	0.00E+00	0.00E+00	3.18E+01	4.00E-03	4.26E-03
7	1.63E+02	1.16E+02	2.78E+02	8.32E+01	8.48E+01
8	1.25E+02	6.89E+01	1.99E+02	3.83E+01	3.15E+01
9	1.18E+03	8.72E-01	6.06E+03	1.03E-01	2.39E+01
10	3.91E+03	3.57E+03	5.20E+03	2.62E+03	2.61E+03
11	2.00E+03	6.72E+01	1.27E+02	1.25E+02	6.08E+01
12	3.82E+06	1.13E+05	5.52E+05	1.29E+03	1.38E+05
13	7.04E+03	1.17E+03	8.47E+02	8.26E+01	3.92E+03
14	9.17E+05	2.47E+04	1.95E+04	5.42E+01	5.96E+02
15	8.56E+03	9.12E+02	1.19E+03	8.27E+01	4.06E+03
16	1.13E+03	9.25E+02	1.24E+03	5.97E+02	3.91E+02
17	7.95E+02	6.62E+02	1.03E+03	5.18E+02	3.34E+02
18	1.46E+06	2.32E+05	2.09E+05	3.01E+01	1.06E+04
19	1.32E+04	3.31E+03	8.67E+03	6.14E+01	6.63E+03
20	6.22E+02	5.40E+02	7.70E+02	2.16E+02	1.67E+02
21	3.39E+02	2.66E+02	4.33E+02	2.40E+02	2.34E+02
22	4.35E+03	3.55E+03	5.97E+03	9.40E+02	4.49E+02
23	5.85E+02	4.91E+02	1.06E+03	4.68E+02	4.50E+02
24	8.49E+02	5.65E+02	1.08E+03	5.27E+02	5.32E+02
25	5.14E+02	5.50E+02	5.41E+02	5.23E+02	5.15E+02
26	1.63E+03	1.83E+03	5.45E+03	1.38E+03	1.49E+03
27	6.61E+02	5.55E+02	1.46E+03	5.98E+02	5.42E+02
28	4.83E+02	4.96E+02	4.89E+02	4.91E+02	4.80E+02
29	9.27E+02	4.96E+02	1.52E+03	6.75E+02	4.77E+02
30	9.04E+05	6.90E+05	7.81E+05	6.14E+05	6.38E+05
Fried.	4.05	2.98	4.20	1.73	2.03
+/=/-	25/4/1	22/0/8	26/0/4	10/5/15	-
W.S.R.	P=0.000	P=0.032	P=0.002	P=0.153	-

6. Conclusion

This study proposes a high-level and adaptive metaheuristic selection-based hybrid algorithm to solve bound-constrained continuous optimization problems. The proposed method contains three important metaheuristics in this subject, namely ABC, DE, and PSO. Adaptive selection process consists of two main stages that are credit assignment and algorithm selection. For the credit assignment, we used the sum of ranks method which

Table 6. The performance of the proposed adaptive metaheuristic selection using two good-performing algorithms for D=50.

Func.	JADE		CMA-ES		Adaptive M.S.
	Mean Err.	T.	Mean Err.	T.	Mean Err.
1	0.00E+00	=	0.00E+00	=	0.00E+00
2	1.56E+13	+	0.00E+00	-	4.52E-02
3	2.41E+06	=	0.00E+00	-	4.45E+05
4	3.55E+03	-	4.34E+03	=	3.91E+03
5	6.75E+03	+	3.48E+03	+	2.71E+03
6	0.00E+00	-	4.00E-01	+	1.02E-03
7	1.20E+04	+	8.32E+03	+	7.61E+03
8	6.67E+03	+	3.83E+03	+	2.92E+03
9	2.03E+03	+	1.03E+01	+	2.48E+00
10	3.64E+05	+	2.62E+05	+	2.29E+05
11	1.77E+04	+	1.25E+04	=	1.09E+04
12	9.74E+05	-	1.29E+05	-	1.81E+06
13	1.88E+05	+	8.26E+03	-	2.93E+04
14	4.14E+06	+	5.42E+03	-	3.11E+04
15	4.72E+04	+	8.27E+03	-	3.14E+04
16	9.70E+04	+	5.97E+04	+	3.05E+04
17	5.99E+04	+	5.18E+04	+	2.05E+04
18	2.87E+06	-	3.01E+03	-	2.45E+05
19	4.31E+04	=	6.14E+03	-	1.66E+04
20	4.45E+04	+	2.16E+04	+	1.05E+04
21	2.63E+04	+	2.40E+04	+	2.31E+04
22	4.09E+05	+	9.40E+04	+	1.00E+04
23	4.89E+04	+	4.68E+04	+	4.41E+04
24	5.57E+04	+	5.27E+04	+	5.18E+04
25	5.31E+04	+	5.23E+04	+	4.77E+04
26	1.90E+05	+	1.38E+05	+	8.71E+04
27	5.11E+04	-	5.98E+04	+	5.05E+04
28	4.98E+04	+	4.91E+04	+	4.66E+04
29	5.07E+04	+	6.75E+04	+	4.68E+04
30	4.21E+07	+	6.14E+07	+	1.52E+07
+/=/-	22/3/5		19/3/8		-

is based on recent performance of the metaheuristics. As for the algorithm selection we used UCB1, which is one of the multiarmed bandit selection algorithms.

The performance of the proposed algorithm was evaluated on CEC'17 benchmark, which includes 30 different test functions. The results obtained suggest that the proposed algorithm outperforms its constituent metaheuristics especially at higher dimensional functions such as 50 and 100. In addition, another comparison was made with some of the important related algorithms in the literature and it is shown that the proposed algorithm is highly comparable with them. Moreover, the case study that includes two good-performing algorithms (namely, CMA-ES and JADE) for the benchmark used in this study, supports the effectiveness of the proposed adaptive selection method.

The main advantage of the proposed algorithm is that it can bring together metaheuristics with different characteristics using an automatic selection of algorithms. Another important advantage is that the proposed method offers a highly generic way to solve problems as including new metaheuristics or excluding existing ones requires less effort than other hybrid algorithms since each metaheuristic works independently from others.

In further research, the effect of other credit assignment and selection methods on the performance of the proposed method can be investigated. Moreover, the proposed method can be tested for different optimization problems with different combinations of metaheuristic algorithms.

Acknowledgment

The first author Osman Gökalp was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) 2211 National Graduate Scholarship Program.

References

[1] Hore S, Chatterjee A, Dewanji A. Improving variable neighborhood search to solve the traveling salesman problem. Applied Soft Computing 2018; 68: 83-91.

[2] Öztürk C, Karaboğa D, Görkemli B. Artificial bee colony algorithm for dynamic deployment of wireless sensor networks. Turkish Journal of Electrical Engineering & Computer Sciences 2012; 20 (2): 255-262.

[3] Yan X, Liu H, Zhu Z, Wu Q. Hybrid genetic algorithm for engineering design problems. Cluster Computing 2017; 20 (1): 263-275.

[4] Beyer HG, Schwefel HP. Evolution strategies–A comprehensive introduction. Natural Computing 2002; 1 (1): 3-52.

[5] Holland JH. Genetic algorithms. Scientific American 1992; 267 (1): 66-73.

[6] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983; 220 (4598): 671-680.

[7] Glover F. Tabu search—part I. ORSA Journal on Computing 1989; 1 (3): 190-206.

[8] Koza JR. Genetic Programming: On The Programming of Computers by Means of Natural Selection. Cambridge, MA, USA: MIT Press, 1992.

[9] Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In: Sixth International Symposium on Micro Machine and Human Science; Nagoya, Japan; 1995. pp. 39-43.

[10] Dorigo M, Di Caro G. Ant colony optimization: a new meta-heuristic. In: IEEE 1999 Congress on Evolutionary Computation; Washington, DC, USA; 1999, pp. 1470-1477.

[11] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1997; 1 (1): 67-82.

[12] Talbi EG. A taxonomy of hybrid metaheuristics. Journal of Heuristics 2002; 8 (5): 541-564.

[13] Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 1997; 11 (4): 341-359.

[14] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization 2007; 39 (3): 459-471.

[15] Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multiarmed bandit problem. Machine Learning 2002; 47 (2-3): 235-256.

[16] Qin AK, Huang VL, Suganthan PN. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Transactions on Evolutionary Computation 2008: 13 (2): 398-417.

[17] Wang H, Wu Z, Rahnamayan S, Sun H, Liu Y et al. Multi-strategy ensemble artificial bee colony algorithm. Information Sciences 2014; 279: 587-603.

[18] Krempser E, Fialho Á, Barbosa HJC. Adaptive operator selection at the hyper-level. In: International Conference on Parallel Problem Solving from Nature; Taormina, Italy; 2012. pp. 378-387.

[19] Awad NH, Ali MZ, Liang JJ, Qu BY, Suganthan PN. Problem Definitions And Evaluation Criteria For The Cec 2017 Special Session And Competition on Single Objective Bound Constrained Real-Parameter Numerical Optimization. Nanyang Technological University: Singapore, 2016.

[20] Fialho Á, Schoenauer M, Sebag M. Toward comparison-based adaptive operator selection. In: The Genetic and Evolutionary Computation Conference; Portland, Oregon, USA; 2010. pp. 767-774.

[21] Zambrano-Bigiarini M, Clerc M, Rojas R. Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. In: IEEE Congress on Evolutionary Computation; Cancun, Mexico; 2013. pp. 2337-2344.

[22] Karaboga D. An idea based on honey bee swarm for numerical optimization. PhD, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[23] López-Ibáñez M, Dubois-Lacoste J, Cáceres LP, Birattari M, Stützle T. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 2016; 3: 43-58.

[24] Karaboga D, Akay B. A comparative study of artificial bee colony algorithm. Applied Mathematics and Computation 2009; 214 (1): 108-132.

[25] Tangherloni A, Rundo L, Nobile MS. Proactive particles in swarm optimization: a settings-free algorithm for real-parameter single objective optimization problems. In: 2017 IEEE Congress on Evolutionary Computation (CEC); San Sebastian, Spain: 2017. pp. 1940-1947.

[26] Hansen N, Müller SD, Koumoutsakos P. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). Evolutionary Computation 2003; 11 (1): 1-18.

[27] Zhang J, Sanderson AC. JADE: adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 2009; 13 (5), 945-958.

[28] Tanabe R, Fukunaga A. Evaluating the performance of SHADE on CEC 2013 benchmark problems. In: IEEE Congress on Evolutionary Computation; Cancun, Mexico; 2013. pp. 1952-1959.

[29] Loshchilov I. CMA-ES with restarts for solving CEC 2013 benchmark problems. In: IEEE Congress on Evolutionary Computation; Cancun, Mexico; 2013. pp. 369-376.