Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 2482543, 23 pages
https://doi.org/10.1155/2019/2482543

Research Article

A New Optimization Algorithm Based on Search and Rescue Operations

Amir Shabani,[1] **Behrouz Asgarian**,[1] **Saeed Asil Gharebaghi**,[1] **Miguel A. Salido**,[2] **and Adriana Giret**[3]

[1]*Faculty of Civil Engineering, K.N.Toosi University of Technology, Tehran, Iran*
[2]*Instituto de Automática e Informática Industrial, Universitat Politècnica de València, Valencia, Spain*
[3]*Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Valencia, Spain*

Correspondence should be addressed to Miguel A. Salido; msalido@dsic.upv.es

Received 20 May 2019; Accepted 3 October 2019; Published 3 November 2019

Academic Editor: Changzhi Wu

In this paper, a new optimization algorithm called the search and rescue optimization algorithm (SAR) is proposed for solving single-objective continuous optimization problems. SAR is inspired by the explorations carried out by humans during search and rescue operations. The performance of SAR was evaluated on fifty-five optimization functions including a set of classic benchmark functions and a set of modern CEC 2013 benchmark functions from the literature. The obtained results were compared with twelve optimization algorithms including well-known optimization algorithms, recent variants of GA, DE, CMA-ES, and PSO, and recent metaheuristic algorithms. The Wilcoxon signed-rank test was used for some of the comparisons, and the convergence behavior of SAR was investigated. The statistical results indicated SAR is highly competitive with the compared algorithms. Also, in order to evaluate the application of SAR on real-world optimization problems, it was applied to three engineering design problems, and the results revealed that SAR is able to find more accurate solutions with fewer function evaluations in comparison with the other existing algorithms. Thus, the proposed algorithm can be considered an efficient optimization method for real-world optimization problems.

1. Introduction

In our world, there are many optimization problems for which different optimization algorithms are used. These algorithms can be classified into deterministic and stochastic optimization algorithms. The deterministic algorithms always produce the same outputs for particular inputs. These algorithms often are used as local search algorithms. Unlike deterministic algorithms, stochastic algorithms have random components and produce different outputs for particular inputs. Many metaheuristic algorithms implement some form of stochastic optimization algorithms [1]. In recent decades, many metaheuristic algorithms have been proposed to solve optimization problems. The genetic algorithm (GA) [2, 3], particle swarm optimization (PSO) [4, 5], and ant colony optimization (ACO) [6, 7] are some of the most widely used metaheuristic algorithms. Some features of these algorithms include simple implementation, flexibility, and capability for finding the local optimum. Most metaheuristic algorithms are inspired by physical or natural phenomena, i.e., animals' movement to find food sources. Consequently, these algorithms are easily understandable and reproducible as software programs for various optimization problems. These algorithms are able to find optimal solutions regardless of the physical nature of the problem. Unlike other optimization methods, metaheuristic algorithms can find global optimal solutions for the problems where there are many local solutions due to their random nature. These reasons have led to extensive use of such algorithms in solving various optimization problems.

In recent years, researchers have carried out extensive studies on metaheuristic algorithms such as harmony search (HS) [8, 9], artificial bee colony (ABC) [10, 11], cuckoo search (CS) [12], imperialist competitive algorithm (ICA)

[13], teaching-learning-based optimization (TLBO) [14], backtracking search optimization algorithm (BSA) [15], firefly algorithm (FA) [16], Yin-Yang-pair optimization (YYPO) [17], and squirrel search algorithm (SSA) [18]. Besides, many metaheuristic algorithms have been enhanced to solve real-world optimization problems such as a decomposition-based multiobjective firefly algorithm developed for RFID network planning [19] and a novel diffusion particle swarm optimization proposed for optimizing sink placement [20]. Based on the "no free lunch" theorem (NFL) [21, 22], there is no optimization algorithm that works well on all optimization problems. An optimization algorithm may achieve very good results on a set of optimization problems, while it is not suitable for others. Therefore, various metaheuristic algorithms might be good for a series of optimization problems, but not for others.

The metaheuristic algorithms can be categorized according to their nature into different groups such as evolution-based, swarm-based, physics-based, and human-based algorithms.

(i) Evolution-based algorithms are developed, based on evolution techniques. The GA, biogeography-based optimizer (BBO) [23, 24], and differential evolution (DE) algorithm [25, 26] are examples of this group of metaheuristic algorithms. For example, the genetic algorithm is inspired by evolution theory.

(ii) In nature, many living beings live socially and search for a variety of goals such as hunting and finding food sources as groups. They use different strategies for searching [27]. Some metaheuristic algorithms solve the optimization problems through modelling the social behavior of living organisms in nature. These types of metaheuristic algorithms are called population-based swarm intelligence (SI) or swarm-based algorithms. Algorithms such as PSO, ABC, ACO, FA, CS, krill herd (KH) [28], simplified dolphin echolocation (SDE) [29], and grey wolf optimizer (GWO) [30] are categorized in this group. For example, PSO has been inspired by movement of organisms in a bird flock or fish collection to search for food sources.

(iii) Physics-based algorithms are inspired by physical phenomena. Algorithms like Big Bang-Big Crunch (BB-BC) [31], colliding bodies optimization (CBO) [32], gravitational search algorithm (GSA) [33, 34], star graph [35], water wave optimization (WWO) [36], and ray optimization [37] are located in this group. For example, WWO is inspired by refraction and breaking rules of water surface waves.

(iv) Human-based algorithms are algorithms that are based on human behavior like tabu search (TS) [38, 39], human mental search (HMS) [40], ICA, and TLBO algorithms. For example, TS and TLBO algorithms have been inspired by human memory function and the way of human learning and training method in the classroom, respectively.

In this paper, a new metaheuristic algorithm is introduced which is based on how to search in search and rescue operations. Humans' search methods have evolved over thousands of years, and there are not any algorithms that used humans' behaviors during this type of search for solving optimization problems. So they encouraged authors to propose a new metaheuristic for solving optimization problems based on these features. This algorithm is categorized as a human-based algorithm. This article is organized as follows: after the introduction in Section 1, Section 2 gives an introduction to search and rescue operations. Section 3 presents the proposed algorithm. In Section 4, comparative tests and benchmarking functions for comparing algorithms are introduced. Section 5 presents the results and discussions, and finally in Section 6, conclusions of this paper are presented.

2. Search and Rescue Operations

Like other living creatures, human beings search for different purposes as groups. Search can be done for a variety of goals such as hunting, finding food sources, or finding lost people. One type of group searches is "search and rescue operations." Search is a systematic operation using available personnel and facilities to locate persons in distress. Rescue is an operation to retrieve persons in distress and deliver them to a safe place [41]. One of the world's earliest search and rescue efforts ensued following the 1656 wreck of the Dutch merchant ship Vergulde Draeck off the west coast of Australia [42].

Search and rescue operations are divided into several types such as mountain rescue, ground search and rescue, urban search and rescue, air-sea rescue, and combat search [43]. In the United States, institutions such as the American Society for Testing and Materials (ASTM) and the National Fire Protection Association (NFPA) provide codes for search and rescue operations [44]. Search and rescue operations are sometimes done to find specific people who are lost. Codes also provide some requirements for searches that increase the chances of successful finding. In the following, the procedure of finding lost people is described considering the main concepts in this operation. Humans can identify the clues and traces of lost people based on the received training. The found clues have different values relative to each other and provide different information about lost people. For example, some clues indicate the likelihood of the presence of lost people in that location. Each group member evaluates clues based on his/her training and delivers this information on found clues through communication equipment to other members. Finally, they can search based on the importance degree of these clues and information that can be obtained from them. Typically, group members search around the clues or seek in directions created by connecting the clues [45]. Therefore, human searches, in search and rescue operations, are divided into two phases: *social* and *individual*. In the social phase, the group members search based on the position of found clues and their quality in areas that are

likely to get better clues. In the individual search phase, the searching is done regardless of the position and amount of clues found by others. Clues can be divided into the following two categories:

(1) Hold clue: one member of the search group is present and searches around it.

(2) Abandoned clue: group members have found the clue and there is no one in that position. In other words, the human who found the clue has left it to find better clues, but the information about that clue is available for group members.

In Figure 1, points A and B are the locations of a group member (human) and a clue, respectively. Path 1, Path 2, and Path 3 are three assumed paths that the lost person has likely passed through. The arrows show the movement directions. In the social phase, the human at position A selects the search direction based on the position of clue B. Since searching around better clues increases the probability of finding the lost person, the area that has better clues in the direction of AB will be selected. In other words, if there are better clues in area 1 compared to area 2, area 1 is chosen; otherwise, area 2 is selected to keep searching. In the sample case depicted in Figure 1, both Path 1 and Path 2 pass through points A and B. If the lost person has passed Path 1 or Path 2, this simple strategy will increase the chances of finding better clues in the social phase. In the individual phase, the human at point A searches around the best clue that is found. This search is done in an area, let us say area 3. If the lost person has passed Path 3, the probability to find him/her is higher during the individual phase compared to the social phase. When a new location is searched by one of these two phases, and the location has better clues than in the previous location (position A), the said location becomes the new position of the group member.

3. A Search and Rescue Optimization Algorithm Proposal

In this section, the mathematical model of the proposed algorithm for solving a "maximization problem" is described. In SAR, the humans' positions are equal to the solutions of the optimization problem, and the amount of clues found in these positions represents the objective function for these solutions. The flowchart of SAR is shown in Figure 2.

3.1. Clues. The group members gather clue information during the search. They left some clues whenever they found better clues in other positions, but information on them are used to improve searching operations. In the model we proposed, the left clues' positions are stored in the memory matrix (matrix M), whereas the humans' positions are stored in a position matrix (matrix X). The dimensions of the matrix M are equal to those of the matrix X. They are $N \times D$ matrices, where D is the dimension of the problem and N is the number of humans. The clues matrix (matrix C) is a matrix containing the positions of found clues. This matrix consists of two matrices X and M. Equation (1) shows how to create C. All new solutions in social and individual phases are created based on the clues matrix, and it is an important part of SAR. The matrices X, M, and C are updated in each human search phase:

$$C = \begin{bmatrix} X \\ M \end{bmatrix} = \begin{bmatrix} X_{11} & \cdots & X_{1D} \\ \vdots & \ddots & \vdots \\ X_{N1} & \cdots & X_{ND} \\ M_{11} & \cdots & M_{1D} \\ \vdots & \ddots & \vdots \\ M_{N1} & \cdots & M_{ND} \end{bmatrix}, \tag{1}$$

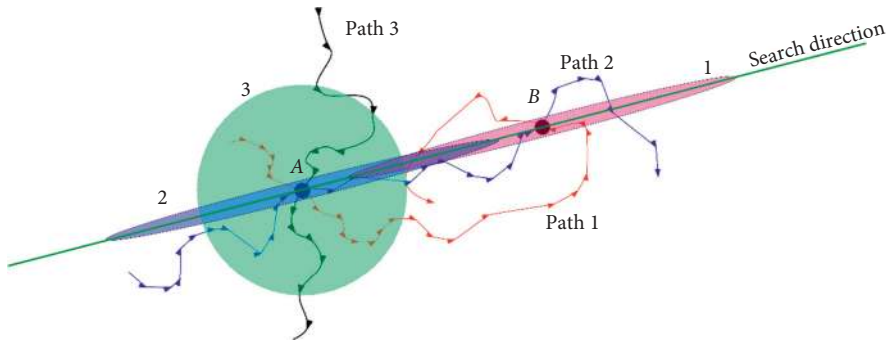
where M and X are memory and humans' position matrices, respectively, and X_{N1} is the position of the 1^{st} dimension for the N^{th} human. Also, M_{1D} is the position of the D^{th} dimension for the 1^{st} memory. The two phases of human searches, including the "social phase" and "individual phase," are modelled as follows.

3.2. Social Phase. Considering the explanations given in the previous section, and taking into account a random clue among found clues, the search direction is obtained using the following equation:

$$\mathrm{SD}_i = (X_i - C_k), \quad k \neq i, \tag{2}$$

where X_i, C_k, and SD_i are the position of the i^{th} human, the position of the k^{th} clue, and the search direction of the i^{th} human, respectively. k is a random integer number ranging between 1 and $2N$ and chosen in a way that $k \neq i$.

It is important to point out that humans normally search in such a way that all desired areas are searched and any repeated location is not searched again. Therefore, the search should be done in a manner that movement of the group members toward each other is limited. To this end, all dimensions of X_i should not be changed by moving in the direction of equation (2). To apply this constraint, the binomial crossover operator has been used. Also as explained in the previous section, if the considered clue is better than the clue related to the current position (the objective function value for solution B is greater than the objective function value for solution A in Figure 1), an area around SD_i direction and around the position of that clue is searched (area 1 in Figure 1); otherwise, the search will continue around the current location along the SD_i direction (area 2 in Figure 1). Finally, the following equation is used for the social phase:



1 and 2: social phase
3: individual phase
A: human position
B: clue position

Figure 1: Two types of human searches in search and rescue operations.

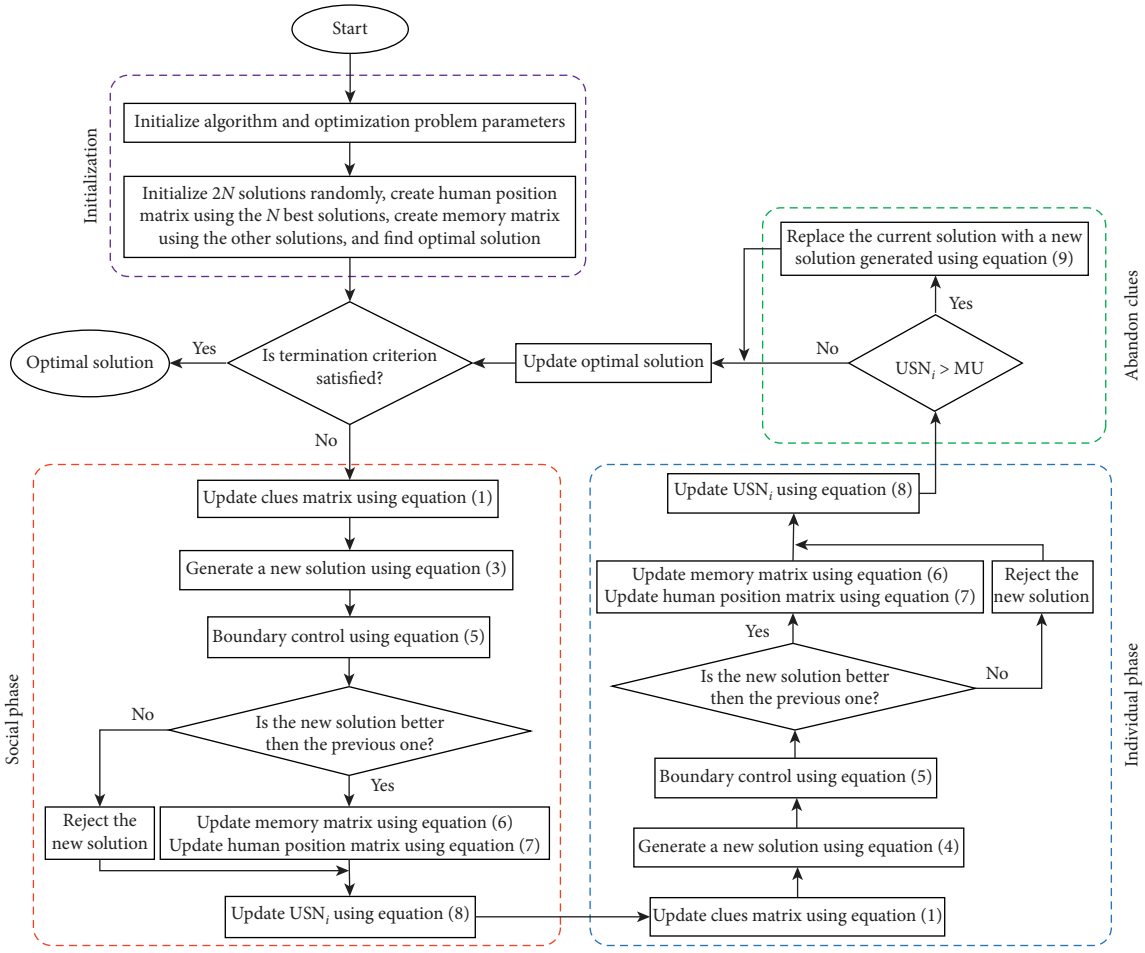


Figure 2: Flowchart of the proposed optimization algorithm (SAR).

$$X'_{i,j} = \begin{cases} \begin{cases} C_{k,j} + r1 \times \left(X_{i,j} - C_{k,j}\right), & \text{if } f\left(C_k\right) > f\left(X_i\right), \\ X_{i,j} + r1 \times \left(X_{i,j} - C_{k,j}\right), & \text{otherwise,} \end{cases} & \text{if } r2 < \text{SE or } j = j_{\text{rand}}, \\ X_{i,j}, & \text{otherwise,} \end{cases} \quad (j = 1, \ldots, D), \tag{3}$$

where $X'_{i,j}$ is the new position of the j^{th} dimension for the i^{th} human; $C_{k,j}$ is the position of the j^{th} dimension for the k^{th} found clue; $f(C_k)$ and $f(X_i)$ are the objective function values for the solutions C_k and X_i, respectively; $r1$ is a random number with a uniform distribution in the range $[-1, 1]$; $r2$ is a uniformly distributed random number in the range $[0, 1]$ and is different for each dimension, but r1 is fixed for all dimensions; j_{rand} is a random integer number ranging between 1 and D which ensures that at least one dimension of $X'_{i,j}$ is different from $X_{i,j}$; and SE is an algorithm parameter ranging between 0 and 1. Equation (3) is used to obtain a new position of the i^{th} human in all dimensions.

3.3. Individual Phase. In the individual phase, humans search around their current position, and the idea of connecting different clues used in the social phase is applied to search. Contrary to the social phase, all dimensions of X_i change in the individual phase. The new position of the i^{th} human is obtained by the following equation:

$$X'_i = X_i + r3 \times (C_k - C_m), \quad i \neq k \neq m, \tag{4}$$

where k and m are random integer numbers ranging between 1 and $2N$. To prevent movement along with other clues, k and m are chosen in such a way that $i \neq k \neq m$. $r3$ is a random number with a uniform distribution ranging between 0 and 1.

3.4. Boundary Control. In all metaheuristic algorithms, all solutions should be located in the solution space, and if they are out of the allowable solution space, they should be modified. So if the new position of a human is out of the solution space, the following equation is used to modify the new position:

$$X'_{i,j} = \begin{cases} \dfrac{\left(X_{i,j} + X_j^{\max}\right)}{2}, & \text{if } X'_{i,j} > X_j^{\max}, \\ \dfrac{\left(X_{i,j} + X_j^{\min}\right)}{2}, & \text{if } X'_{i,j} < X_j^{\min}, \end{cases} \quad (j = 1, \ldots, D), \tag{5}$$

where $X_j^{\max}$ and $X_j^{\min}$ are the values of the maximum and minimum threshold for the j^{th} dimension, respectively.

3.5. Updating Information and Positions. In each iteration, the group members will search according to these two phases, and after each phase, if the value of objective function in position $X'_i(f(X'_i))$ is greater than the previous one $(f(X_i))$, the previous position (X_i) will be stored in a random position of the memory matrix (M) using equation (6) and this position will be accepted as a new position using equation (7). Otherwise, this position is left and the memory is not updated:

$$M_n = \begin{cases} X_i, & \text{if } f(X'_i) > f(X_i), \\ M_n, & \text{otherwise}, \end{cases} \tag{6}$$

$$X_i = \begin{cases} X'_i, & \text{if } f(X'_i) > f(X_i), \\ X_i, & \text{otherwise}, \end{cases} \tag{7}$$

where M_n is the position of the n^{th} stored clue in the memory matrix and n is a random integer number ranging between 1 and N. Using this type of memory updating increases the diversity of the algorithm and the ability of the algorithm to find the global optimum as well.

3.6. Abandoning Clues. In search and rescue operations, time is a very important factor because the lost people may be injured and the delay of search and rescue teams may result in their deaths. Therefore, these operations must be done in such a way that the largest space is searched in the shortest possible time. So if a human cannot find better clues after a certain number of searches around his/her current position, he/she leaves the current position and goes to a new position. To model this behavior, at first, unsuccessful search number (USN) is set to 0 for each human being. Whenever a human finds better clues in the first or second phase of the search, the USN is set to 0 for that human; otherwise, it will increase by 1 point as presented in the following equation:

$$\text{USN}_i = \begin{cases} \text{USN}_i + 1, & \text{if } f(X'_i) < f(X_i), \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

where USN_i indicates the number of times the human i has not been able to find better clues. When the USN for a human is greater than the maximum unsuccessful search number (MU), he/she goes to a random position in the search space using equation (9), and the USN_i is set to 0 for that human:

$$X_{i,j} = X_j^{\min} + r4 \times \left(X_j^{\max} - X_j^{\min}\right), \quad j = 1, \ldots, D, \tag{9}$$

where $r4$ is a random number with a uniform distribution ranging between 0 and 1. It is different for each dimension.

3.7. Control Parameters of SAR. SAR has two control parameters: SE (social effect) and MU (maximum unsuccessful search number). The SE is used to control the effect of group members on each other in the social phase. This parameter ranges within [0, 1]. Greater values of the SE increase the convergence rate and also decrease the global search ability of the algorithms. The MU parameter indicates the maximum number of unsuccessful searches before leaving a clue. It ranges within $[0, 2 \times T_{\max}]$, where $2 \times T_{\max}$ is the maximum number of searches done by each human and $T_{\max}$ is the maximum number of iterations. For greater values of the MU, humans will never leave the clues. On the one hand, small values of this parameter lead to the group 3 member finish searching around the current clue and go to other locations before he/she can completely search around it. On the other hand, large values of this parameter cause an

increase in searches around one clue and a reduction of the chances of searching in other regions. The MU directly relates to the dimension of the problem. As the search space increases, the maximum number of unsuccessful searches is increased, too.

For all the following tests, the value of the SE was set to 0.05 and the value of the MU was obtained by equation (10). The analysis of SAR parameters has shown that these values for the SE and MU are suitable for solving single-objective continuous optimization problems.

$$\mathrm{MU} = 70 \times D. \tag{10}$$

3.8. Pseudocode of Search and Rescue Optimization Algorithm (SAR). The pseudocode of this algorithm is presented in Algorithm 1 for solving a maximization problem. Position sorting is performed only once before the iterations begin.

3.9. Conceptual Comparison of SAR with Other Metaheuristic Algorithms. In each iteration of SAR, two new solutions are generated in two phases for each algorithm particle. The concepts of searching in these phases are different. In the social phase, new solutions are generated based on locations and objective functions of other solutions. In this phase, each particle can move toward the other particles. But in the individual phase, new solutions are generated around current solutions and each particle does not move toward the other particles. In this algorithm, only a new solution which is better than the current solution is accepted and the current solution is stored in an archive (memory). The archive is used to generate new solutions.

SAR such as PSO, ABC, DE, GSA, and TLBO is a population-based optimization algorithm. Similarity and difference of SAR and these algorithms are explained as follows.

3.9.1. SAR versus PSO

Similarity. Both of them utilize social and individual information and memory to generate new solutions.

Difference. In PSO, a combination of the global best position (social information) and local best position (individual information) is used to generate a new solution. But SAR separately uses social and individual information to generate two new solutions. Besides, the global best solution is not considered by SAR in these phases. PSO accepts all new solutions, but SAR only accepts new solutions which are better than current solutions. Unlike PSO, SAR leaves unimproved solutions. Also, SAR considers objective function values to generate new solutions in the social phase. Besides, the memory update mechanisms of these methods are different.

3.9.2. SAR versus ABC

Similarity. To produce a new solution, objective function values are considered by both of these algorithms. They only accept a new solution which is better than the current solution. Both of them leave unimproved solutions.

Difference. ABC has not any kinds of memories. It selects solutions by the roulette wheel mechanism and generates new solutions by changing only one dimension of the selected solutions. But SAR selects clues randomly and uses them to generate new solutions by changing some or all dimensions of current solutions. So they use different strategies to generate new solutions, and SAR produces two new solutions for each agent in each iteration.

3.9.3. SAR versus DE

Similarity. They accept only new solutions which are better than current solutions. The crossover mechanism used in the social phase of SAR is similar to the crossover mechanism of DE.

Difference. DE does not consider the previous solutions to generate new solutions, and it is a memoryless algorithm. Although both of these algorithms utilize the same crossover mechanism, the equations applied to generate new solutions are different. DE does not leave unimproved solutions and does not consider objective functions values to produce new solutions. Also, DE produces only a new solution for each agent in each iteration.

3.9.4. SAR versus TLBO

Similarity. Both of them include two searching phases and consider values of objective function to generate new solutions. They accept only new solutions which are better than current solutions.

Difference. The previous solutions are not used by TLBO, and it is a memoryless algorithm. Unlike this algorithm, SAR leaves unimproved solutions after a certain number of unsuccessful objective function evaluations. SAR and TLBO obtain new solutions using entirely different strategies.

Furthermore, in comparison with the above algorithms, the boundary control strategy of SAR is different.

3.10. Computational Complexity. In this section, the computational complexity of SAR is discussed. The population initialization process requires $O(2 \times n \times d)$ times, where n and d indicate the number of humans and the dimension of the problem. The proposed algorithm requires $O(2n \cdot \log(2n))$ times to sort population in the initialization phase. The complexity of the social and individual phases is $O(n \times d)$ times in the worst case. The complexity of the abandon clue process is $O(n)$ and $O(n \times d)$ in the best and the worst case. Thus, the computational complexity of SAR is as follows:

(1) **Begin:**
(2) Randomly initialize a population of $2N$ solutions uniformly distributed in the range $[X_j^{\min}, X_j^{\max}]$, $j = 1, \ldots, D$
(3) Sort the solutions in the decreasing order and find the best position (Xbest)
(4) Use the first half of the sorted solutions for human position matrix (X) and the others for memory matrix (M)
(5) Define the algorithm parameters (SE, MU) and set $\mathrm{USN_i} = 0$ where $i = 1, \ldots, N$
(6) **While** stop criterion is not satisfied **do**
(7) **For** $i = 1$ to N **do**

Social phase

(8) $$C = \begin{bmatrix} X \\ M \end{bmatrix}$$
(9) $\mathrm{SD}_i = (X_i - C_k)$, k is randomly selected in such a way that $i \neq k$
(10) $j_{\mathrm{rand}} = \mathrm{rand\,int}[1, D]$
(11) $r1 = \mathrm{rand}[-1, 1]$
(12) **For** $j = 1$ to D **do**
(13) $$X'_{i,j} = \begin{cases} \begin{cases} C_{k,j} + r1 \times SD_{i,j}, & \text{if } f(C_k) > f(X_i), \\ X_{i,j} + r1 \times SD_{i,j}, & \text{otherwise} \end{cases} & \text{if } \mathrm{rand}[0,1] < \mathrm{SE} \text{ or } j = j_{\mathrm{rand}} \\ X_{i,j}, & \text{otherwise} \end{cases}$$
(14) $$X'_{i,j} = \begin{cases} (X_{i,j} + X_j^{\max})/2, & \text{if } X'_{i,j} > X_j^{\max} \\ (X_{i,j} + X_j^{\min})/2, & \text{if } X'_{i,j} < X_j^{\min} \end{cases}$$
(15) **End For**
(16) $$M_n = \begin{cases} X_i, & \text{if } f(X'_i) > f(X_i), \\ M_n, & \text{otherwise}, \end{cases} \quad n \text{ randomly selected}$$
(17) $$X_i = \begin{cases} X'_i, & \text{if } f(X'_i) > f(X_i) \\ X_i, & \text{otherwise} \end{cases}$$
(18) $$\mathrm{USN}_i = \begin{cases} \mathrm{USN}_i + 1, & \text{if } f(X'_i) < f(X_i) \\ 0, & \text{otherwise} \end{cases}$$

Individual phase

(19) $$C = \begin{bmatrix} X \\ M \end{bmatrix}$$
(20) $X'_i = X_i + \mathrm{rand}[0,1] \times (C_k - C_m)$, k and m are randomly selected in such a way that $i \neq k \neq m$
(21) **For** $j = 1$ to D **do**
(22) $$X'_{i,j} = \begin{cases} (X_{i,j} + X_j^{\max})/2, & \text{if } X'_{i,j} > X_j^{\max} \\ (X_{i,j} + X_j^{\min})/2, & \text{if } X'_{i,j} < X_j^{\min} \end{cases}$$
(23) **End For**
(24) $$M_n = \begin{cases} X_i, & \text{if } f(X'_i) > f(X_i), \\ M_n, & \text{otherwise}, \end{cases} \quad n \text{ randomly selected}$$
(25) $$X_i = \begin{cases} X'_i, & \text{if } f(X'_i) > f(X_i) \\ X_i, & \text{otherwise} \end{cases}$$
(26) $$\mathrm{USN}_i = \begin{cases} \mathrm{USN}_i + 1, & \text{if } f(X'_i) < f(X_i) \\ 0, & \text{otherwise} \end{cases}$$
(27) **If** $\mathrm{USN}_i > \mathrm{MU}$ **do**
(28) **For** $j = 1$ to D **do**
(29) $X_{i,j} = X_j^{\min} + \mathrm{rand}[0,1] \times (X_j^{\max} - X_j^{\min})$
(30) **End for**
(31) $\mathrm{USN}_i = 0$
(32) **End If**
(33) **End for**
(34) Find the current best position and update Xbest
(35) **End while**
(36) Return Xbest
(37) **End**

Algorithm 1: Pseudocode of the search and rescue optimization algorithm (SAR).

$$\begin{aligned} O(\text{SAR}) &= O(\text{initialization}) + \text{Maxit} \times (O(\text{social phase}) \\ &\quad + O(\text{individual phase}) + O(\text{abandon clue})) \\ &= O(2 \times n \times d) + O(2n \log(2n)) + \text{Maxit} \\ &\quad \times (O(n \times d) + O(n \times d) + O(n \times d)), \end{aligned} \tag{11}$$

where Maxit is the maximum number of iterations. According to the above discussion, the total computational complexity of SAR is $O(\text{Maxit} \times n \times d)$.

4. Numerical Results

To evaluate the performance of SAR, four tests were considered. Classic benchmark functions were used in test 1 and test 2, while modern benchmarks have been used in test 3. Finally, some structural engineering design problems were utilized in test 4. All runs were executed on a 64 bit computer with 32 GB of RAM having an Intel i7 (3.4 GHz) CPU running Windows 10.

4.1. Test 1. As discussed in Introduction, metaheuristic algorithms are inherently divided into four groups. One well-known algorithm was selected from each group to compare with the proposed algorithm. Artificial bee colony (ABC), gravitational search algorithm (GSA), differential evolution (DE), and teaching-learning-based optimization (TLBO) algorithms were considered swarm-based, physics-based, evolution-based, and human-based algorithms, respectively. The population size and control parameters of these algorithms are given in Table 1, as suggested in [46], [33], [25], and [47]. TLBO has no control parameter.

In the first test, the performance of SAR was compared with that of ABC, GSA, DE, and TLBO on 27 benchmark functions. These functions used by various researchers [16, 48–50] are presented in Table 2. In this table, type, D, range, and $f_{\min}$ represent the type of the benchmark function, the dimensions of the problem, the range of variations, and the optimal value of the function, respectively. Also, $X_j^{\max}$ and $X_j^{\min}$ are the maximum and minimum threshold values of the dimension j, respectively. In the type column of this table, U, M, S, and N refer to unimodal, multimodal, separable, and nonseparable functions, respectively. 13 benchmark functions are unimodal, while 14 functions are multimodal. Moreover, there are 11 separable functions and 16 nonseparable functions. Since in some of these classical functions the locations of global minima are symmetrical, some algorithms may be affected by this feature and show a different and unrealistic performance. Therefore, the locations of symmetrical global minima are shifted using function T defined in the last row of Table 2. This function generates nonsymmetrical numbers.

In test 1, the maximum number of function evaluations (NFE) was set to $4 \times 10^3 \times D$ for all algorithms, in which D is the number of dimensions of the benchmark function that is specified in Table 2. All algorithms were independently executed 51 times. The algorithms were stopped when the number of evaluations for the objective function exceeded the NFE ($4 \times 10^3 \times D$) or when the least error (distance between the objective function of the best found solution and the objective function of the global optimum solution) was less than 10^{-8}.

Table 1: Control parameters of the compared algorithms in tests 1 and 2.

Algorithm	Control parameters	Population size
ABC	Limit = 50 D	50
GSA	$G0 = 50$, $\alpha = 20$, and $T = 50$	50
DE	$F = 0.5$ and CR = 0.9	20
TLBO	—	20
SAR	SE = 0.05 and MU = 70 D	20

4.2. Test 2. The algorithms and benchmark functions considered for this test are similar to those in the first test. The difference between the first and second tests is related to NFEs. For all of these benchmark functions (except for $f13$), this value is equal to $2 \times 10^4 \times D$ which is 5 times greater than the value considered in the first test. For $f13$, the NFE is set to $3 \times 10^4 \times D$. The purpose of the second test is to examine the ability of the algorithms to find global minima. Therefore, a high value of the NFE is considered.

As in the first test, all the algorithms were independently run 51 times, and the algorithms were stopped when the error values (distance between the objective function value of the best found solution and the objective function value of the global optimum solution) for the global solution were less than 10^{-8} or the number of function evaluations reached its maximum. The population and parameters of the algorithms were set the same as those in test 1.

4.3. Test 3. In this test, 28 benchmark functions of CEC 2013 Competition on Single-Objective Real-Parameter Numerical Optimization are used to compare SAR with the state-of-the-art optimization algorithms. All of these functions are minimization problems. The details of CEC 2013 benchmark functions can be found in [51]. These functions cover various types of optimization problems. These functions are divided into three classes: unimodal (C1–C5), basic multimodal (C6–C20), and composition (C21–C28) benchmark functions. The composition functions are created by combinations of different basic functions. Consequently, they are multimodal, nonseparable, and asymmetrical. The algorithm was independently run 51 times, and it stops when the number of function evaluations reached 100,000 or when the error values from the global optima were less than 10^{-8}. The control parameters of SAR were the same as those in the previous two tests. The problem dimension was 10, and the variables ranged within [−100, 100]. To verify the performance of SAR on these problems, it was compared with nine optimization algorithms. These algorithms which were verified on CEC 2013 benchmark functions are listed as follows:

(i) Artificial bee colony (ABC)

(ii) A CMA-ES super-fit scheme for the resampled inheritance search (CMA-RIS) [52]

Table 2: Benchmark functions used in tests 1 and 2.

Test function	Type	D	Range	$f_{\min}$
$f_1(x) = 25 + \sum_{i=1}^{n} \lfloor x_i \rfloor$	US	5	$[-5.12, 5.12]$	-5
$f_2(x) = \sum_{i=1}^{n} y_i^2,\ y_i = x_i - T_i$	US	25	$[-100, 100]$	0
$f_3(x) = \sum_{i=1}^{n} (y_i + 0.5)^2,\quad y_i = x_i - T_i$	US	30	$[-100, 100]$	0
$f_4(x) = \sum_{i=1}^{n} i y_i^2,\ y_i = x_i - T_i$	US	30	$[-10, 10]$	0
$f_5(x) = \sum_{i=1}^{n} \lvert y_i \rvert^{i+1},\ y_i = x_i - T_i$	US	30	$[-1, 1]$	0
$f_6(x) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$	UN	2	$[-4.5, 4.5]$	0
$f_7(x) = 0.26(y_1^2 + y_2^2)^2 - 0.48 y_1 y_2,\ y_i = x_i - T_i$	UN	2	$[-10, 10]$	0
$f_8(x) = 100(y_1^2 - y_2)^2 + (y_1 - 1)^2 + (y_3 - 1)^2 + 90(y_3^2 - y_4)^2 + 10.1((y_2 - 1)^2 + (y_4 - 1)^2) + 19.8(y_2 - 1)(y_4 - 1),\ y_i = x_i - T_i$	UN	4	$[-10, 10]$	0
$f_9(x) = \sum_{i=1}^{n} y_i^2 + (\sum_{i=1}^{n} 0.5 i y_i^2)^2 + (\sum_{i=1}^{n} 0.5 i y_i^2)^4,\ y_i = x_i - 0.5T_i$	UN	10	$[-5, 10]$	0
$f_{10}(x) = \sum_{i=1}^{n/4} (y_{4i-3} + 10 y_{4i-2})^2 + 5(y_{4i-1} - y_{4i})^2 + (y_{4i-2} - y_{4i-1})^4 + 10(y_{4i-3} - y_{4i})^4,\ y_i = x_i - 0.8T_i$	UN	24	$[-4, 5]$	0
$f_{11}(x) = \sum_{i=1}^{n} \lvert y_i \rvert + \prod_{i=1}^{n} \lvert y_i \rvert,\ y_i = x_i - T_i$	UN	30	$[-10, 10]$	0
$f_{12}(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} y_j)^2,\ y_i = x_i - T_i$	UN	10	$[-100, 100]$	0
$f_{13}(x) = \sum_{i=1}^{n-1} [100(y_{i+1} - y_i^2)^2 + (y_i - 1)^2],\ y_i = x_i - 0.5T_i$	UN	25	$[-30, 30]$	0
$f_{14}(x) = (x_2 - (5.1/4\pi)x_1^2 + (5/\pi)x_1 - 6)^2 + 10(1 - (1/8\pi))\cos x_1 + 10$	MS	2	$-5 \le x_1 \le 10,\ 0 \le x_2 \le 15$	0.3978873577
$f_{15}(x) = y_1^2 + 2y_2^2 - 0.3\cos(3\pi y_1) - 0.4\cos(4\pi y_2) + 0.7,\ y_i = x_i - T_i$	MS	2	$[-100, 100]$	0
$f_{16}(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(i x_i^2/\pi))^{2m},\ m = 10$	MS	2	$[0, \pi]$	-1.8013034101
$f_{17}(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(i x_i^2/\pi))^{2m},\ m = 10$	MS	5	$[0, \pi]$	-4.6876581791
$f_{18}(x) = \sum_{i=1}^{n} [y_i^2 - 10\cos(2\pi y_i) + 10],\ y_i = x_i - 0.5T_i$	MS	25	$[-5.12, 5.12]$	0
$f_{19}(x) = \sum_{i=1}^{n} -y_i \sin(\sqrt{\lvert y_i \rvert}),\ y_i = x_i - 0.05T_i$	MS	25	$[-500, 500]$	$-418.98288727 * D$
$f_{20}(x) = y_1^2 + 2y_2^2 - 0.3\cos(3\pi y_1)\cos(4\pi y_2) + 0.3,\ y_i = x_i - T_i$	MN	2	$[-100, 100]$	0
$f_{21}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2)]$	MN	2	$[-2, 2]$	3
$f_{22}(x) = (\sum_{i=1}^{5} i\cos((i+1)x_1 + i))(\sum_{i=1}^{5} i\cos((i+1)x_2 + i))$	MN	2	$[-10, 10]$	-186.73090883
$f_{23}(x) = 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	MN	2	$[-5, 5]$	-1.0316284535
$f_{24}(x) = -20\exp(-0.2\sqrt{(1/n)\sum_{i=1}^{n} y_i^2}) - \exp((1/n)\sum_{i=1}^{n}\cos(2\pi y_i)) + 20 + e,\ y_i = x_i - T_i$	MN	25	$[-32, 32]$	0
$f_{25}(x) = (1/4000)\sum_{i=1}^{n} y_i^2 - \prod_{i=1}^{n}\cos(y_i/\sqrt{i}) + 1,\ y_i = x_i - T_i$	MN	25	$[-600, 600]$	0
$f_{26}(x) = (\pi/n)\{10\sin^2(\pi z_1) + \sum_{i=1}^{n-} 1(z_i - 1)^2[1 + \sin^2(\pi z_{i+1})] + (z_n - 1)^2\} + \sum_{i=1}^{n} u(y_i, 10, 100, 4),\ (z_i = 1 + 0.25(y_i + 1),\ y_i = x_i - T_i) u(y_i, a, k, m) = \begin{cases} k(y_i - a)^m, & y_i > a \\ 0, & -a \le y_i \le a \\ k(-y_i - a)^m, & y_i < -a \end{cases}$	MN	30	$[-50, 50]$	0
$f_{27}(x) = 0.1\{\sin^2(3\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + \sin^2(3\pi y_{i+1})] + (y_n - 1)^2[1 + \sin^2(2\pi y_n)]\} + \sum_{i=1}^{n} u(y_i, 5, 100, 4),\ y_i = x_i - T_i$	MN	30	$[-50, 50]$	0
$T_i = 0.25(X_i^{\max} - X_i^{\min})((-1)^{(\cos(j(\pi/4))+2)})(\sin(j(3\pi/D) + 2)^3 + \cos(j(4\pi/D) - 1)^3)$	—	—	—	—

(iii) Adaptive monogamous pairs genetic algorithm (AMopGA) [53]

(iv) Grey wolf optimizer (GWO)

(v) Yin-Yang-pair optimization (YYPO) [17]

(vi) Reflected adaptive differential evolution with two external archives (RJADE) [54]

(vii) Self-adaptive differential evolution (SaDE) [55]

(viii) Self-adaptive heterogeneous PSO (f_k-PSO) [56]

(ix) Standard Particle Swarm Optimisation 2011 (SPSO) [57]

These algorithms include a CMA variant (CMA-RIS), a GA variant (AMopGA), two DE variants (SaDE and RJADE), two PSO variants (f_k-PSO and SPSO), two recent metaheuristic algorithms (GWO and YYPO), and ABC. Different studies show that the performances of these variants are better than that of the basic version of them. The results of ABC, GWO, and YYPO are reported in [17]. For

the other above algorithms, the results from the original studies were used.

4.4. Test 4. In order to evaluate the performance of SAR for solving real-world engineering optimization problems, three engineering design problems were utilized. The penalty function approach was employed to handle the constraints of these problems. For all the engineering optimization problems, the control parameters of SAR were set the same as those in the previous tests and the population size of SAR was equal to 10. SAR was independently run 50 times. These engineering problems are introduced as follows.

4.4.1. I-Beam Design. The I-beam design problem was firstly proposed by Gold and Krishnamurty [58]. The objective is to find the minimum vertical deflection of an I-beam. The vertical deflection of an I-beam defined by equation (12) is dependent on design load (P), length of the beam (L), and modulus of elasticity (E):

$$f(x) = \frac{PL^3}{48EI}. \tag{12}$$

P, L, and E are 600 kN, 200 cm, and 20000 kN/cm^2. This problem has four variables and two constraints including the cross-sectional area (constraint g_1) and the bending stress of the beam (constraint g_2). The maximum cross-sectional area is 300 cm^2, and the allowable bending stress of the beam is 56 kN/cm^2. The mathematical model of the problem is expressed as follows:

$$\begin{aligned}
&\text{minimize} \quad f\left(h, b, t_w, t_f\right) = \frac{5000}{\left(t_w\left(h - 2t_f\right)^3/12\right) + \left(bt_f^3/6\right) + \left(2bt_f\left(h - t_f/2\right)^2\right)},\\
&\text{subject to} \quad g_1(x) = 2bt_f + t_w\left(h - 2t_f\right) \le 300,\\
&\qquad\qquad\ g_2(x) = \frac{1.8 \times 10^5 h}{t_w\left(h - 2t_f\right)^3 + 2bt_w\left(4t_f^2 + 3h\left(h - 2t_f\right)\right)} + \frac{1.5 \times 10^4 b}{t_w^3\left(h - 2t_f\right) + 2t_w b^3} \le 56,
\end{aligned} \tag{13}$$

with bounds $10 \le h \le 80$, $10 \le b \le 50$, $0.9 \le t_f \le 5$, and $0.9 \le t_w \le 5$.

4.4.2. Cantilever Beam Design. The cantilever beam design problem was firstly presented by Fleury and Braibant [59]. It consists of five hollow square cross sections. The thickness of these sections is fixed, and the height of them is a design variable. A vertical force is applied at the free end of the beam, and another end of the beam is a rigid support.

The cantilever beam design problem has five variables and one constraint. The goal is to minimize the weight of the beam. This problem can be stated as follows:

$$\begin{aligned}
&\text{minimize } f(x) = 0.0624\left(x_1 + x_2 + x_3 + x_4 + x_5\right),\\
&\text{subject to } g(x) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} \le 1,
\end{aligned} \tag{14}$$

with bounds $0.01 \le x_i \le 100$, $i = 1, 2, \ldots, 5$.

4.4.3. Spatial 25-Bar Truss Structure Design. The spatial 25-bar truss design was widely used in structural design optimization. Many optimization methods were applied to solve this well-known optimization problem. The elastic modulus and the material density of all members are 10^4 ksi and 0.1 lb/in^3, respectively. The minimum and maximum cross-sectional areas of them are 0.01 in^2 and 3.4 in^2, respectively. This truss is subjected to the two loading conditions presented in Table 3.

Because of the symmetry of the structure, the 25 members of the truss are divided into 8 groups, as follows: (1) $A1$, (2) $A2$–$A5$, (3) $A6$–$A9$, (4) $A10$-$A11$, (5) $A12$-$A13$, (6) $A14$–$A17$, (7) $A18$–$A21$, and (8) $A22$–$A25$. The displacements of the nodes in both directions are limited to ± 0.35 in, and the allowable stress for each group is shown in Table 4.

5. Results and Discussion

5.1. Performance of SAR on Classic Benchmark Functions. The means and variances of errors (distance between the minimum value of found objective functions and the optimal value of the function) obtained by the algorithms for the first test are shown in Table 5. Errors less than 10^{-8} are considered 0.

The Wilcoxon signed-rank test was used to compare the pair algorithms, and the results of 51 runs for SAR were compared with those of the other algorithms [60]. In the Wilcoxon signed-rank test, the superiority of the two algorithms is seen using the hypothesis test. Two hypotheses are defined as null hypothesis (H0) and alternative hypothesis (H1). The null hypothesis indicates that there is no

Table 3: Nodal loading (ksi) for the spatial 25-bar truss.

Node	Case 1			Case 2		
	P_x	P_y	P_z	P_x	P_y	P_z
1	0	20	−5	1	10	−5
2	0	−20	−5	0	10	−5
3	0	0	0	0.5	0	0
6	0	0	0	0.5	0	0

Table 4: Member stress limitations (ksi) for the spatial 25-bar truss.

Design variables	Members	Compression
A_1	35.092	40
A_2–A_5	11.59	40
A_6–A_9	17.305	40
A_{10}-A_{11}	35.092	40
A_{12}-A_{13}	35.092	40
A_{14}–A_{17}	6.759	40
A_{18}–A_{21}	6.959	40
A_{22}–A_{25}	11.082	40

Table 5: Basic statistical results of 51 runs obtained by ABC, DE, GSA, TLBO, and SAR in test 1.

Function	ABC		DE		GSA		TLBO		SAR	
	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.	Mean	Std.
f_1	0	0	0	0	$4.00E+00$	$3.46E-01$	0	0	0	0
f_2	0	0	0	0	0	0	0	0	0	0
f_3	0	0	$3.33E-01$	$8.41E-01$	0	0	$2.73E+03$	$2.40E+03$	0	0
f_4	0	0	0	0	0	0	0	0	0	0
f_5	0	0	$1.20E-08$	$1.80E-08$	0	0	0	0	0	0
f_6	$1.22E-03$	$1.50E-03$	0	0	$2.41E-03$	$1.02E-03$	0	0	0	0
f_7	$6.31E-04$	$7.91E-04$	0	0	$4.02E-05$	$2.61E-04$	0	0	0	0
f_8	$3.00E-01$	$2.33E-01$	$3.50E-02$	$1.44E-01$	$1.97E+00$	$1.98E+00$	$9.55E-03$	$6.40E-02$	$1.45E-06$	$8.87E-06$
f_9	$7.30E+00$	$3.98E+00$	0	0	$3.31E-01$	$6.48E-01$	0	0	0	0
f_{10}	$2.19E-02$	$6.46E-03$	$2.60E-05$	$2.24E-05$	$5.82E-04$	$2.60E-04$	$5.43E-04$	$6.39E-04$	$6.30E-08$	$1.28E-07$
f_{11}	$1.58E-08$	0	0	0	$1.80E-08$	0	$1.55E-06$	$7.55E-06$	0	0
f_{12}	$1.56E+02$	$1.03E+02$	0	0	$9.40E+01$	$7.27E+01$	0	0	0	0
f_{13}	$4.08E-01$	$5.24E-01$	$2.12E+01$	$1.39E+01$	$2.25E+01$	$1.60E+01$	$1.03E+02$	$4.83E+02$	$2.73E+00$	$2.55E+00$
f_{14}	$3.83E-07$	$1.96E-06$	0	0	0	0	0	0	0	0
f_{15}	0	0	0	0	0	0	0	0	0	0
f_{16}	0	0	0	0	0	0	0	0	0	0
f_{17}	$1.55E-08$	$4.72E-08$	$9.01E-03$	$1.73E-02$	$1.13E-01$	$9.05E-02$	$1.16E-01$	$8.01E-02$	0	0
f_{18}	$2.24E-08$	$8.22E-08$	$3.44E+01$	$2.66E+01$	$2.31E+01$	$4.46E+00$	$8.14E+01$	$1.91E+01$	$9.75E-02$	$2.99E-01$
f_{19}	$1.32E+02$	$6.01E+01$	$1.41E+03$	$3.46E+02$	$8.08E+03$	$3.51E+02$	$3.18E+03$	$5.73E+02$	$2.63E+00$	$1.67E+01$
f_{20}	0	0	0	0	0	0	0	0	0	0
f_{21}	$1.77E-05$	$1.00E-04$	0	0	0	0	0	0	0	0
f_{22}	$1.41E-04$	$3.67E-04$	$2.35E-04$	$6.21E-04$	$1.04E+01$	$1.23E+01$	$1.78E-04$	$8.72E-04$	$3.36E-07$	$1.04E-06$
f_{23}	0	0	0	0	0	0	0	0	0	0
f_{24}	$1.42E-06$	$1.01E-06$	0	0	0	0	$1.20E+01$	$2.47E+00$	0	0
f_{25}	$3.79E-04$	$1.58E-03$	$1.50E-03$	$3.72E-03$	$1.74E+02$	$2.26E+01$	$1.11E+00$	$4.49E+00$	0	0
f_{26}	0	0	$6.10E-03$	$2.46E-02$	$5.31E-01$	$7.39E-01$	$7.33E+00$	$7.29E+00$	0	0
f_{27}	0	0	$5.98E+02$	$4.27E+03$	0	0	$4.93E+00$	$6.90E+00$	0	0

difference between two algorithms, and the alternative hypothesis indicates that there is a difference between two algorithms. To determine whether these algorithms have any superiority to each other, the p value is used. The smaller the p value is, the more likely these two algorithms are different (alternative hypothesis). To determine the level of hypothesis through the level of significance, α has been used. The Wilcoxon signed-rank test results for the first test are shown in Table 6. In this paper, α is equal to 0.05. If the p value is less than 0.05, then these two algorithms are statistically different with the 95% confidence level. When h is 0, it means that there is no difference between the two

Table 6: Comparisons between SAR and the other algorithms by the Wilcoxon signed-rank test ($\alpha = 0.05$).

Function	ABC		DE		GSA		TLBO	
	p value	h	p value	h	p value	h	p value	h
f_1	$1.00E+00$	0	$1.00E+00$	0	$5.12E+12$	1^+	$1.00E+00$	0
f_2	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0
f_3	$1.00E+00$	0	$4.88E+04$	1^+	$1.00E+00$	0	$2.65E+10$	1^+
f_4	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0
f_5	$1.00E+00$	0	$6.25E+02$	0	$1.00E+00$	0	$1.00E+00$	0
f_6	$2.65E-10$	1^+	$1.00E+00$	0	$2.65E+10$	1^+	$1.00E+00$	0
f_7	$2.65E+10$	1^+	$1.00E+00$	0	$1.56E+02$	1^+	$1.00E+00$	0
f_8	$2.65E+10$	1^+	$2.70E+03$	1^+	$2.65E+10$	1^+	$2.03E+09$	1^+
f_9	$2.65E+10$	1^+	$1.00E+00$	0	$1.91E+07$	1^+	$1.00E+00$	0
f_{10}	$2.65E+10$	1^+	$2.65E+10$	1^+	$2.65E+10$	1^+	$2.65E+10$	1^+
f_{11}	$3.94E+09$	1^+	$1.00E+00$	0	$2.65E+10$	1^+	$9.77E+04$	1^+
f_{12}	$2.65E+10$	1^+	$1.00E+00$	0	$2.65E+10$	1^+	$1.00E+00$	0
f_{13}	$2.51E+05$	1^+	$2.81E+10$	1^+	$2.65E+10$	1^+	$1.21E+09$	1^+
f_{14}	$4.17E+07$	1^+	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0
f_{15}	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0
f_{16}	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0
f_{17}	$3.13E+02$	1^+	$4.88E+04$	1^+	$3.05E+09$	1^+	$6.48E+10$	1^+
f_{18}	$1.07E+01$	0	$2.65E+10$	1^+	$2.63E+10$	1^+	$2.65E+10$	1^+
f_{19}	$2.65E+10$	1^+	$2.65E+10$	1^+	$2.65E+10$	1^+	$2.65E+10$	1^+
f_{20}	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0
f_{21}	$5.80E+09$	1^+	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0
f_{22}	$2.65E+10$	1^+	$1.48E+07$	1^+	$2.22E+09$	1^+	$4.53E+02$	1^+
f_{23}	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0	$1.00E+00$	0
f_{24}	$2.65E+10$	1^+	$1.00E+00$	0	$1.00E+00$	0	$2.65E+10$	1^+
f_{25}	$6.25E+02$	0	$3.90E+03$	1^+	$2.65E+10$	1^+	$2.65E+10$	1^+
f_{26}	$1.00E+00$	0	$1.25E+01$	0	$4.16E+07$	1^+	$2.65E+10$	1^+
f_{27}	$1.00E+00$	0	$3.13E+02$	1^+	$1.00E+00$	0	$5.73E+10$	1^+
+/=/−	**13/13/1**		**9/17/1**		**15/12/0**		**13/14/0**	

algorithms. 1^+ for h indicates significant superiority of the first algorithm over the second one. Finally, 1^- for h implies a significant superiority of the second algorithm over the first one.

According to the data in Table 5, SAR and ABC found the solution for all five US benchmarks—both unimodal and separable—in all runs. But the TLBO, DE, and GSA failed to find the global minimum in 2, 2, and one functions, respectively. The DE and SAR maintained almost the same performance on 8 UN functions, and they found the global minimum point on all runs except for three functions ($f8$, $f10$, and $f13$) that the performance of SAR was better than that of DE. TLBO had a performance near to that of these two algorithms. But it failed to find the minimum value in $f11$ in addition to these three functions. Both ABC and GSA failed to find the minimum value for any UN functions and had less convergence rate for these functions compared to the other algorithms. SAR had the best performance on MS functions and was able to find the value of global minima in four of six functions. Like the other algorithms, it failed to find the global minima in $f18$ and $f19$. Among the other four algorithms, ABC had less mean of errors. On 8 MN functions, SAR performed better than the other algorithms, and it only failed to find the global minimum in $f22$, but it had less errors than the others.

In Table 6, SAR was compared with the other algorithms by the two-sided Wilcoxon signed-rank test. In the last row of this table, the superiority of the first algorithm (SAR) over the second one has been shown by the sign (+). Equal superiority is shown with the sign (=). Finally, the superiority of the second algorithm over the first one is shown with the sign (−). According to the data in Table 6, SAR outperformed ABC in 13 functions, while they had similar performance in 13 functions. ABC outperformed SAR only in $f13$. TLBO and GSA were not better than SAR in any functions. DE outperformed SAR in $f8$, while SAR outperformed it in 9 functions, and they had similar performance in 17 functions. Accordingly, SAR outperformed ABC, DE, GSA, and TLBO in the classic benchmark functions.

5.2. Global Search Ability. In the second test, the ability of the algorithms to find the global minimum has been investigated, and for this purpose, the maximum number of function evaluations (NFE) was increased. In Table 7, the success percentages of the algorithms in finding the global minimum for 27 benchmark functions along with the average percentages are shown for 4 benchmark function types (unimodal, multimodal, separable, and nonseparable functions).

The least NFE among 5 algorithms is highlighted in bold for each function. ABC was not able to find the minimum value of UN functions for any runs except for $f11$. This fact indicates the slower convergence rate of this algorithm to solve these types of optimization problems. Moreover, as it is clear in Table 7, ABC had the lowest success rate for such problems among all the algorithms, while it performed

Table 7: Success percentage and mean number of function evaluations for ABC, DE, GSA, TLBO, and SAR.

Function	ABC		DE		GSA		TLBO		SAR	
	Success %	Mean eval.	Success %	Mean eval.	Success %	Mean eval.	Success %	Mean eval.	Success %	Mean eval.
f_1	100	1265	100	744	0	1*e*4	100	**453**	100	1762
f_2	100	62338	100	**28474**	100	270315	100	51351	100	30252
f_3	100	20459	84	105482	100	50556	0	6*e*5	100	**16334**
f_4	100	65100	100	**28689**	100	327167	100	63975	100	30777
f_5	100	24281	94	43030	100	118365	100	21029	100	**7706**
f_6	0	4*e*4	100	**1649**	100	22650	100	2302	100	3279
f_7	0	4*e*4	100	1408	100	14054	100	**1119**	100	1948
f_8	0	8*e*4	94	**13116**	0	8*e*4	100	30940	100	16711
f_9	0	2*e*5	100	15023	100	109991	100	**12527**	100	20680
f_{10}	0	4.8*e*5	24	464170	0	4.8*e*5	0	4.8*e*5	100	**107834**
f_{11}	100	121765	100	52864	4	599957	100	103706	100	**51684**
f_{12}	0	2*e*5	100	**18450**	100	108097	100	23064	100	25306
f_{13}	0	7.5*e*5	2	7.38*e*5	0	7.5*e*5	0	7.5*e*5	100	**156253**
f_{14}	100	8998	100	2488	100	16660	100	**1759**	100	3458
f_{15}	100	4170	100	2213	100	18721	100	**1367**	100	2156
f_{16}	100	2747	100	1637	100	18902	100	**1020**	100	1498
f_{17}	100	15600	73	33279	14	94383	20	83000	100	**8634**
f_{18}	100	92589	0	5*e*5	0	5*e*5	0	5*e*5	100	**86189**
f_{19}	100	173076	0	5*e*5	0	5*e*5	0	5*e*5	100	**98887**
f_{20}	100	5922	100	2221	100	18651	100	**1526**	100	2449
f_{21}	98	11425	100	1817	100	21524	100	**1325**	100	2566
f_{22}	100	13495	100	9219	0	4*e*4	100	8136	100	**7428**
f_{23}	100	4373	100	2217	100	17268	100	**1336**	100	2134
f_{24}	100	123414	98	53792	100	472456	0	5*e*5	100	**48492**
f_{25}	98	91770	78	131265	80	284346	0	5*e*5	100	**38684**
f_{26}	100	69014	96	64814	98	261425	8	561692	100	**35346**
f_{27}	100	76217	96	70921	100	293661	4	582169	100	**38992**
U fun. avg.	46.2		84.6		61.8		76.9		100	
M fun. avg.	99.7		81.5		70.9		52.2		100	
S fun. avg.	100		77.4		64.9		65.4		100	
N fun. avg.	56.0		86.9		67.6		63.2		100	
Total average	73.9		83.0		66.5		64.1		100	

better after SAR for multimodal functions. It also had a great performance for separable problems. DE had the best performance for nonseparable functions after SAR. Regarding the average success percentage of the algorithms in finding the global minimum, SAR had better performance than ABC, DE, GSA, and TLBO. This algorithm was able to find the global minimum for multimodal functions with an average of 100% indicating high ability of it for global searching and avoiding local minima.

5.3. Convergence Rate Analysis. In Table 7, in addition to the success percentages, the average numbers of function evaluations to reach stopping conditions are presented for the compared algorithms. Based on this table, SAR had the fastest convergence rate for 13 functions among all studied algorithms. In Figure 3, the convergence curves of SAR for some of the benchmark functions are shown. These functions include unimodal and separable ($f2$, $f3$, and $f4$), unimodal and nonseparable ($f6$, $f7$, and $f10$), multimodal and separable ($f15$, $f17$, and $f19$), and multimodal and nonseparable ($f20$, $f22$, and $f25$) functions. These curves were obtained by averaging 51 independent runs for all the algorithms.

Also, the Wilcoxon signed-rank test has been used to more accurately check the convergence rate of the algorithms. In this test, the maximum numbers of function evaluations of SAR are compared with those of the other algorithms in 51 runs by the two-sided Wilcoxon signed-rank test ($\alpha = 0.05$), and the results are presented in Table 8. The comparison procedure is similar to that in Table 6. 1^+ for h indicates the first algorithm has a faster convergence rate than the second algorithm.

According to the data in Table 8, SAR had a faster convergence rate than ABC in 26 functions, while ABC had a faster convergence rate than SAR only in $f1$. Therefore, it can be concluded that SAR has a faster convergence rate than ABC. Also, the convergence rate of SAR was faster than that of GSA in all 27 functions. The convergence rate of SAR was faster than that of DE in 10 functions, while it had lower convergence rate than DE in 13 functions, and they had equal convergence rate in 4 functions. In the unimodal functions, the convergence rate of DE was faster than that of SAR. But SAR had a faster convergence rate in the multimodal functions. In comparison with TLBO, the convergence rate of SAR was faster, equal, and lower in 15, 4, and 11 functions, respectively.

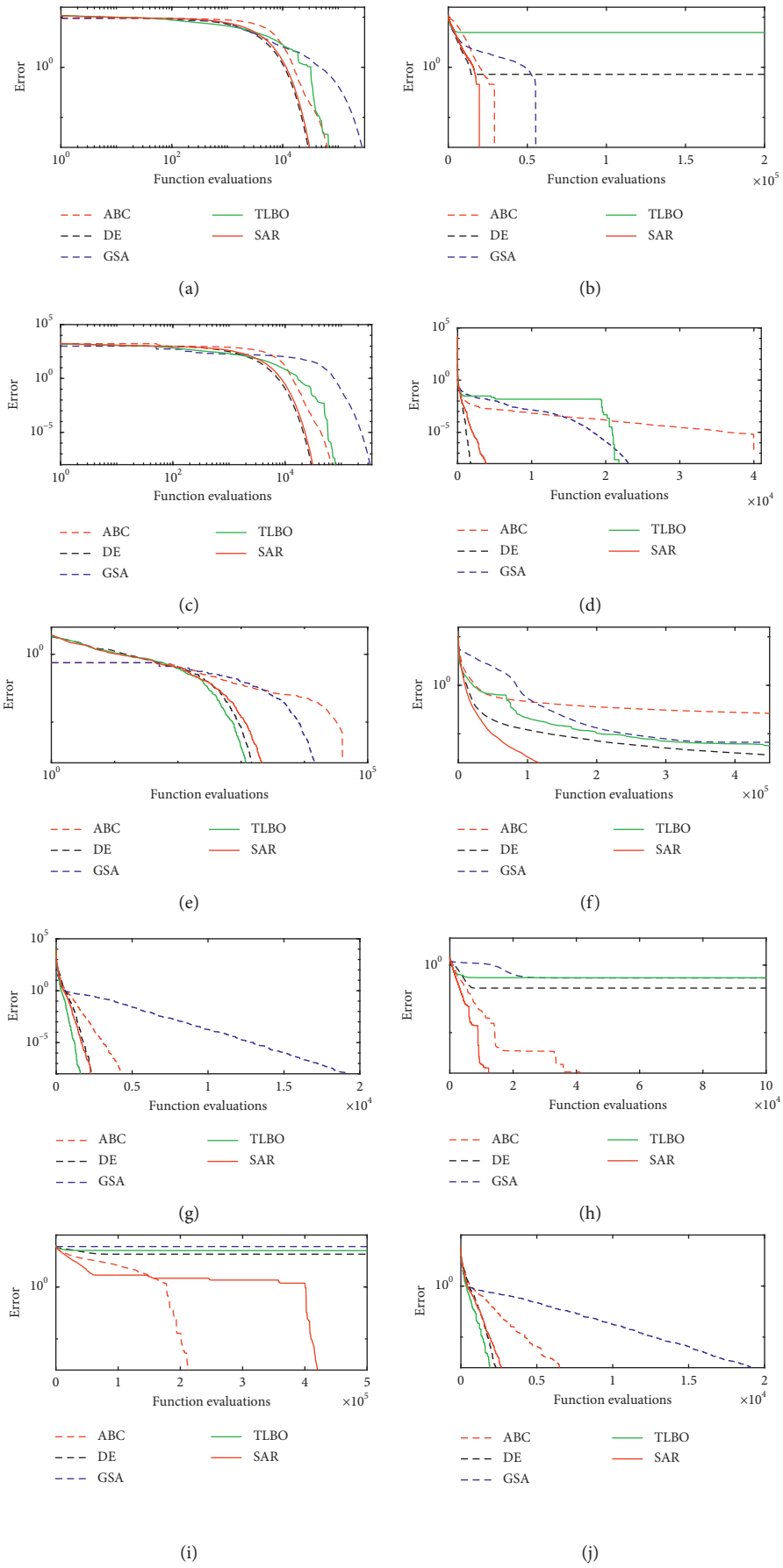


Figure 3: Continued.

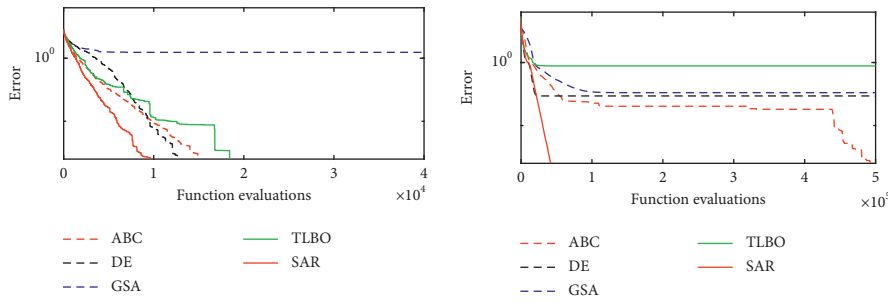


Figure 3: Convergence curves of SAR and the compared algorithms. (a) $F2$. (b) $F3$. (c) $F4$. (d) $F6$. (e) $F7$. (f) $F10$. (g) $F15$. (h) $F17$. (i) $F19$. (j) $F20$. (k) $F22$. (l). $F25$.

Table 8: Comparisons of NFEs between SAR and the other algorithms by the Wilcoxon signed-rank test ($\alpha = 0.05$).

Function	ABC		DE		GSA		TLBO	
	p value	h	p value	h	p value	h	p value	h
f_1	$4.38E-05$	1^-	$3.57E-10$	1^-	$2.65E-10$	1^+	$2.65E-10$	1^-
f_2	$2.65E-10$	1^+	$8.10E-10$	1^-	$2.65E-10$	1^+	$2.65E-10$	1^+
f_3	$3.79E-10$	1^+	$4.16E-03$	1^-	$2.65E-10$	1^+	$2.65E-10$	1^+
f_4	$2.65E-10$	1^+	$6.54E-09$	1^-	$2.65E-10$	1^+	$2.65E-10$	1^+
f_5	$2.65E-10$	1^+	$1.42E-01$	0	$2.65E-10$	1^+	$2.65E-10$	1^+
f_6	$2.65E-10$	1^+	$2.65E-10$	1^-	$2.65E-10$	1^+	$8.25E-08$	1^-
f_7	$2.65E-10$	1^+	$6.42E-10$	1^-	$2.65E-10$	1^+	$2.65E-10$	1^-
f_8	$2.65E-10$	1^+	$5.52E-05$	1^-	$2.65E-10$	1^+	$2.81E-10$	1^+
f_9	$2.65E-10$	1^+	$2.65E-10$	1^-	$2.65E-10$	1^+	$2.65E-10$	1^-
f_{10}	$2.65E-10$	1^+	$2.65E-10$	1^+	$2.65E-10$	1^+	$2.65E-10$	1^+
f_{11}	$2.65E-10$	1^+	$3.20E-05$	1^+	$2.65E-10$	1^+	$2.65E-10$	1^+
f_{12}	$2.65E-10$	1^+	$2.65E-10$	1^-	$2.65E-10$	1^+	$5.05E-06$	1^-
f_{13}	$2.65E-10$	1^+	$2.65E-10$	1^+	$2.65E-10$	1^+	$2.65E-10$	1^+
f_{14}	$2.65E-10$	1^+	$1.23E-07$	1^-	$2.65E-10$	1^+	$5.60E-10$	1^-
f_{15}	$2.65E-10$	1^+	$3.01E-02$	1^+	$2.65E-10$	1^+	$2.65E-10$	1^-
f_{16}	$2.65E-10$	1^+	$8.86E-04$	1^+	$2.65E-10$	1^+	$4.52E-10$	1^-
f_{17}	$2.65E-10$	1^+	$6.87E-02$	0	$2.65E-10$	1^+	$7.22E-10$	1^+
f_{18}	$4.05E-03$	1^+	$2.65E-10$	1^+	$2.65E-10$	1^+	$2.65E-10$	1^+
f_{19}	$2.13E-07$	1^+	$2.65E-10$	1^+	$2.65E-10$	1^+	$2.65E-10$	1^+
f_{20}	$2.65E-10$	1^+	$1.26E-05$	1^-	$2.65E-10$	1^+	$2.65E-10$	1^-
f_{21}	$2.65E-10$	1^+	$2.99E-10$	1^-	$2.65E-10$	1^+	$2.65E-10$	1^-
f_{22}	$2.81E-10$	1^+	$1.70E-06$	1^+	$2.65E-10$	1^+	$4.20E-01$	0
f_{23}	$2.65E-10$	1^+	$7.78E-01$	0	$2.65E-10$	1^+	$3.78E-10$	1^-
f_{24}	$2.65E-10$	1^+	$5.55E-09$	1^-	$2.65E-10$	1^+	$2.65E-10$	1^+
f_{25}	$2.65E-10$	1^+	$7.12E-02$	0	$2.65E-10$	1^+	$2.65E-10$	1^+
f_{26}	$2.65E-10$	1^+	$6.40E-06$	1^+	$2.65E-10$	1^+	$2.65E-10$	1^+
f_{27}	$4.97E-09$	1^+	$1.06E-05$	1^+	$2.65E-10$	1^+	$2.65E-10$	1^+
+/=/−	**26/0/1**		**10/4/13**		**27/0/0**		**15/1/11**	

According to the data in Table 8 and Figure 3, the convergence rate of SAR was significantly higher than that of GSA and ABC. Also, SAR had a faster convergence rate compared to TLBO in more than half of the functions. The convergence rate of DE was almost equal to that of SAR. But SAR had a faster convergence rate than DE for multimodal functions.

5.4. Performance of SAR on CEC 2013 Benchmark Function Set. The mean of errors obtained by ABC, CMA-RIS, AMopGA, GWO, YYPO, RJADE, SaDE, f_k-PSO, SPSO, and SAR algorithms for the third test is reported in Tables 9–11. The solutions less than 10^{-8} are considered 0. Moreover, by comparing these 10 algorithms, the rank of each algorithm is determined for each function and is provided in these tables. The lowest rank (1) is related to an algorithm with the lowest mean of errors compared to the other algorithms. The best mean of errors is highlighted in bold in these tables.

In Table 9, the results obtained by the algorithms for unimodal functions are shown. These functions are very suitable for comparing the ability of the algorithms in

Table 9: Results of CEC 2013 unimodal benchmark functions for the compared algorithms.

Function		ABC	CMA-RIS	AMopGA	GWO	YYPO	RJADE	SaDE	f_k-PSO	SPSO	SAR
$C1$	Mean	**0.00E+00**	**0.00E+00**	$3.14E-06$	$3.48E+00$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Rank	1	1	9	10	1	1	1	1	1	1
$C2$	Mean	$2.47E+06$	**0.00E+00**	**0.00E+00**	$1.12E+06$	$6.44E+04$	**0.00E+00**	$1.66E-03$	$1.44E+05$	$3.63E+04$	**0.00E+00**
	Rank	10	1	1	9	7	1	5	8	6	**1**
$C3$	Mean	$9.94E+06$	$7.04E-01$	**0.00E+00**	$9.49E+07$	$7.43E+05$	$1.21E+02$	$1.24E+01$	$6.75E+05$	$2.68E+05$	$2.30E+00$
	Rank	9	2	1	10	8	5	4	7	6	3
$C4$	Mean	$9.24E+03$	**0.00E+00**	$6.80E-04$	$7.02E+03$	$1.01E+02$	$1.16E+02$	$1.83E-04$	$4.16E+02$	$8.87E+03$	**0.00E+00**
	Rank	10	1	4	8	5	6	3	7	9	1
$C5$	Mean	**0.00E+00**	**0.00E+00**	$1.08E-04$	$2.39E+01$	$1.08E-05$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Rank	1	1	9	10	8	1	1	1	1	1

Table 10: Results of CEC 2013 basic multimodal benchmark functions for the compared algorithms.

Function		ABC	CMA-RIS	AMopGA	GWO	YYPO	RJADE	SaDE	f_k-PSO	SPSO	SAR
$C6$	Mean	$3.02E-01$	$1.10E+00$	$6.70E+00$	$2.21E+01$	$6.26E+00$	$7.89E+00$	$5.77E+00$	$2.64E+00$	$9.80E+00$	**1.76E−01**
	Rank	2	3	7	10	6	8	5	4	9	**1**
$C7$	Mean	$3.29E+01$	$5.33E+01$	**0.00E+00**	$8.38E+00$	$4.19E+00$	$1.59E-01$	$1.30E-01$	$1.92E+00$	$2.11E+01$	$7.27E+00$
	Rank	9	10	**1**	7	5	3	2	4	8	6
$C8$	Mean	$2.04E+01$	$2.03E+01$	**1.45E+01**	$2.04E+01$	$2.03E+01$	$2.04E+01$	$2.04E+01$	$2.03E+01$	$2.03E+01$	$2.04E+01$
	Rank	6	2	**1**	6	2	6	6	2	2	6
$C9$	Mean	$5.02E+00$	$3.59E+00$	**8.36E−02**	$3.57E+00$	$3.07E+00$	$4.46E+00$	$1.40E+00$	$2.75E+00$	$4.80E+00$	$3.44E+00$
	Rank	10	7	**1**	6	4	8	2	3	9	5
$C10$	Mean	$1.43E+00$	**1.24E−02**	$5.59E-01$	$9.55E+00$	$2.36E-01$	$3.53E-02$	$2.28E-02$	$5.13E-01$	$3.00E-01$	$3.41E-02$
	Rank	9	**1**	8	10	5	4	2	7	6	3
$C11$	Mean	**0.00E+00**	$3.57E+00$	**0.00E+00**	$9.98E+00$	$2.97E-03$	**0.00E+00**	**0.00E+00**	$1.76E-01$	$1.09E+01$	**0.00E+00**
	Rank	**1**	8	**1**	9	6	**1**	**1**	7	10	**1**
$C12$	Mean	$2.23E+01$	$1.29E+01$	$2.31E+01$	$1.64E+01$	$1.98E+01$	$7.72E+00$	**4.48E+00**	$7.04E+00$	$1.39E+01$	$1.08E+01$
	Rank	9	5	10	7	8	3	**1**	2	6	4
$C13$	Mean	$3.05E+01$	$2.56E+01$	$6.60E+00$	$1.99E+01$	$2.47E+01$	$6.76E+00$	**5.47E+00**	$1.15E+01$	$2.08E+01$	$1.67E+01$
	Rank	10	9	2	6	8	3	**1**	4	7	5
$C14$	Mean	$1.24E-01$	$1.02E+02$	$6.95E+01$	$4.59E+02$	$1.38E+00$	**1.20E−02**	$1.59E-02$	$3.78E+01$	$8.34E+02$	$1.15E-01$
	Rank	4	8	7	9	5	**1**	2	6	10	3
$C15$	Mean	$6.84E+02$	$6.17E+02$	$1.17E+03$	$6.24E+02$	$6.47E+02$	$6.67E+02$	$6.86E+02$	**4.54E+02**	$7.74E+02$	$6.02E+02$
	Rank	7	3	10	4	5	6	8	**1**	9	2
$C16$	Mean	$7.79E-01$	**1.64E−01**	$8.48E-01$	$1.15E+00$	$8.17E-01$	$1.13E+00$	$1.12E+00$	$4.07E-01$	$5.00E-01$	$5.24E-01$
	Rank	5	**1**	7	10	6	9	8	2	3	4
$C17$	Mean	$8.57E+00$	$1.04E+01$	$1.33E+01$	$2.59E+01$	$9.55E+00$	$1.01E+01$	$1.01E+01$	$1.10E+01$	$1.89E+01$	**7.18E+00**
	Rank	2	6	8	10	3	4	4	7	9	**1**
$C18$	Mean	$3.97E+01$	$2.98E+01$	$8.43E+01$	$3.67E+01$	$2.50E+01$	$2.27E+01$	$2.28E+01$	**1.56E+01**	$1.78E+01$	$2.18E+01$
	Rank	9	7	10	8	6	4	5	**1**	2	3
$C19$	Mean	**4.27E−02**	$8.14E-01$	$8.14E+00$	$1.56E+00$	$4.49E-01$	$4.42E-01$	$3.76E-01$	$5.01E-01$	$9.00E-01$	$1.99E-01$
	Rank	**1**	7	10	9	5	4	3	6	8	2
$C20$	Mean	$3.24E+00$	$4.16E+00$	$2.50E+00$	$2.59E+00$	$2.53E+00$	$2.53E+00$	**2.23E+00**	$2.52E+00$	$3.40E+00$	$2.77E+00$
	Rank	8	10	2	6	4	4	**1**	3	9	7

exploitation. As it is clear from Table 9, SAR and CMA-RIS were able to find 100% of the minimum point for all runs in 4 cases of these 5 functions and are better than the other algorithms. The mean error of CMA-RIS is lower than that of SAR on the $C3$ function. Therefore, CMA-RIS is the best algorithm for unimodal functions. The mean ranks and the overall ranks of them are shown in Table 12. Regarding the data in this table, after CMA-RIS, SAR is the second best algorithm among 10 compared algorithms for CEC 2013 unimodal functions. The DE variants are the third best algorithms, and there is a high difference between mean ranks of them (mean rank: 2.8) and those of SAR and CMA-RIS (mean rank: 1.2 and 1.4, respectively). Hence, SAR is efficient for unimodal functions, and it has high exploitation capability.

Table 10 illustrates the results of the algorithms for basic multimodal functions. These functions are very suitable for comparing the ability of the algorithms in exploration and finding the global optimum. SaDE and AMopGA found the best results for 4 of 15 basic multimodal functions. According to the data in Table 12, SaDE was the first best algorithm with mean rank 3.4 and SAR was ranked the second with mean rank 3.53. The performance of SaDE is slightly better than that of SAR. After them, f_k-PSO with

Table 11: Results of CEC 2013 composition benchmark functions for the compared algorithms.

Function		ABC	CMA-RIS	AMopGA	GWO	YYPO	RJADE	SaDE	f_k-PSO	SPSO	SAR
$C21$	Mean	**1.33E + 02**	$1.61E+02$	$3.06E+02$	$3.93E+02$	$3.73E+02$	$3.96E+02$	$3.96E+02$	$3.75E+02$	$4.00E+02$	$1.98E+02$
	Rank	1	2	4	7	5	8	8	6	10	3
$C22$	Mean	$1.23E+01$	$2.44E+02$	$2.15E+02$	$5.27E+02$	$8.68E+01$	$2.70E+01$	$1.13E+01$	$1.22E+02$	$9.06E+02$	**1.12E + 01**
	Rank	3	8	7	9	5	4	2	6	10	1
$C23$	Mean	$1.01E+03$	$8.35E+02$	**2.21E + 02**	$7.26E+02$	$9.14E+02$	$7.00E+02$	$6.55E+02$	$5.15E+02$	$9.10E+02$	$8.41E+02$
	Rank	10	6	1	5	9	4	3	2	8	7
$C24$	Mean	$1.33E+02$	**1.19E + 02**	$1.37E+02$	$2.10E+02$	$1.86E+02$	$2.02E+02$	$1.94E+02$	$2.03E+02$	$2.14E+02$	$1.25E+02$
	Rank	3	1	4	9	5	7	6	8	10	2
$C25$	Mean	**1.56E + 02**	$1.93E+02$	$1.90E+02$	$2.10E+02$	$2.00E+02$	$2.03E+02$	$1.98E+02$	$2.05E+02$	$2.09E+02$	$1.92E+02$
	Rank	1	4	2	10	6	7	5	8	9	3
$C26$	Mean	$1.30E+02$	$1.61E+02$	$2.39E+02$	$2.06E+02$	$1.31E+02$	$1.27E+02$	$1.27E+02$	$1.89E+02$	$2.00E+02$	**1.12E + 02**
	Rank	4	6	10	9	5	2	2	7	8	1
$C27$	Mean	$3.87E+02$	$3.13E+02$	$4.94E+02$	$4.20E+02$	$3.36E+02$	$3.04E+02$	**3.00E + 02**	$3.70E+02$	$3.36E+02$	$3.52E+02$
	Rank	8	3	10	9	4	2	1	7	4	6
$C28$	Mean	**1.30E + 02**	$2.06E+02$	$3.50E+02$	$3.50E+02$	$2.86E+02$	$2.88E+02$	$2.96E+02$	$3.26E+02$	$3.00E+02$	$2.22E+02$
	Rank	1	2	9	9	4	5	6	8	7	3

mean rank 3.93 was the third best algorithm among the 10 algorithms that have been compared for basic multimodal functions. These results confirmed a higher exploration ability of SAR in comparison with the other algorithms.

The results obtained by the 10 algorithms compared on 8 composition functions are shown in Table 11. Among all the compared algorithms, SAR obtained the best results for $C22$ and $C26$ functions and ABC for $C21$, $C25$, and $C28$ functions. From Table 12, it can be seen that SAR with mean rank 3.25 had the best performance compared to the other algorithms on the composition functions. After SAR, ABC was the second best algorithm with mean rank 3.88 and CMA-RIS was located at the next rank with mean rank 4 among the 10 algorithms. The results show that SAR outperformed the compared algorithms for these kinds of optimization problems.

In Table 12, means and overall ranks of the compared algorithms on unimodal, multimodal, and composition benchmark functions of CEC 2013 are reported. Furthermore, means and overall ranks of these algorithms on all the benchmark functions are presented in this table. It can be seen that the mean rank of SAR is the lowest, and it is the best algorithm among all the studied algorithms on the CEC 2013 benchmark function set. After SAR, the performance of SaDE is better than that of the others. It slightly outperformed CMA-RIS. RJADE was the fourth best algorithm in this test. The performance of GWO is the worst for all kinds of benchmark functions, and it was ranked 10th among the 10 studied algorithms.

In Tables 9–12, the algorithms were compared and ranked in the group. In Table 13, the mean of errors obtained by each of these 9 algorithms was compared with that of SAR.

According to the data in Table 13, SAR was significantly better than ABC, GWO, YYPO, AMopGA, RJADE, and SPSO. After SAR, SaDE, CMA-RIS, and RJADE have achieved the best performance, respectively. SaDE and CMA-RIS had less mean of errors than SAR in 9 functions, while SAR had less errors than them in 15 functions, and the mean of errors of SAR and them was equal in 4 functions.

5.5. Application of SAR on Engineering Optimization Problems. SAR was applied on three engineering design problems, and the obtained results were compared with those of other algorithms documented in the literature. These problems have been defined in the previous section. SAR and all the compared algorithms satisfied the constraints of these problems. In the following tables, "Std.," "NFE," and "NA" mean standard deviation, number of function evaluations, and not available, respectively, and the best results are highlighted in bold.

5.5.1. I-Beam Design. This problem was solved using several other methods including cuckoo search (CS) [61], adaptive response surface method (ARSM) [62], improved ARSM (IARSM) [62], and symbiotic organisms search (SOS) [63]. The results obtained by SAR and the mentioned algorithms are presented in Table 14.

The optimal designs found by SAR and SOS are the same and better than those by the others. The average and standard deviation of SAR are slightly better than those of SOS. The convergence curve of SAR for the I-beam design problem is shown in Figure 4.

5.5.2. Cantilever Beam Design. Table 15 compares the optimal designs found by SAR and the other algorithms including the method of moving asymptotes (MMA) [64], generalized convex approximation I and II (GCA(I) and GCA(II)) [64], symbiotic organisms search (SOS) [63], cuckoo search (CS) [61], flower pollination algorithm (FPA) [65], and modified firefly algorithm (MFA) [66]. According to the data in this table, the lowest cantilever beam weight was found by SAR. Moreover, SAR only required 10000 function evaluations to find the optimum design, and it is less than that of the others. Hence, SAR had the fastest convergence rate among all the compared algorithms. Also, the average and standard deviation of SAR are better than those of SOS.

The convergence curve of SAR for the cantilever beam design problem is shown in Figure 5. These comparative results show SAR outperforms all the studied algorithms for this design problem.

Table 12: Ranks of the compared algorithms on CEC 2013 benchmark functions.

Algorithm	Unimodal		Basic multimodal		Composition		All the functions	
	Mean rank	Overall rank	Mean rank	Overall rank	Mean rank	Overall rank	Mean rank	Overall rank
ABC	6.20	9	6.13	8	3.88	2	5.40	6
CMA-RIS	1.20	1	5.80	7	4.00	3	3.67	3
AMopGA	4.80	6	5.67	6	5.88	7	5.45	7
GWO	9.40	10	7.80	10	8.38	10	8.53	10
YYPO	5.80	8	5.20	5	5.38	6	5.46	8
RJADE	2.80	3	4.53	4	4.88	5	4.07	4
SaDE	2.80	3	3.40	1	4.13	4	3.44	2
f_k-PSO	4.80	6	3.93	3	6.50	8	5.08	5
SPSO	4.60	5	7.13	9	8.25	9	6.66	9
SAR	1.40	2	3.53	2	3.25	1	2.73	1

Table 13: Results of comparison between SAR and the other algorithms for all CEC 2013 benchmark functions.

		ABC	CMA-RIS	AMopGA	GWO	YYPO	RJADE	SaDE	f_k-PSO	SPSO
SAR	+	21	15	18	26	22	16	15	16	22
	=	3	4	2	0	1	5	4	2	2
	−	4	9	8	2	5	7	9	10	4

Table 14: Comparison results for the I-beam design.

Variables	ARSM	IARSM	CS	SOS	SAR
H	80	79.99	80	80	80
B	37.05	48.42	50	50	50
t_w	1.71	0.9	0.9	0.9	0.9
t_f	2.31	2.4	2.321672	2.32179	2.32179
$f_{\min}$	0.0157	0.0131	0.013075	**0.013074**	**0.013074**
Average	NA	NA	0.013536	0.013088	**0.013084**
Std.	NA	NA	$1.30E-04$	$4.00E-05$	**2.40E − 05**
NFE	NA	NA	5000	5000	5000

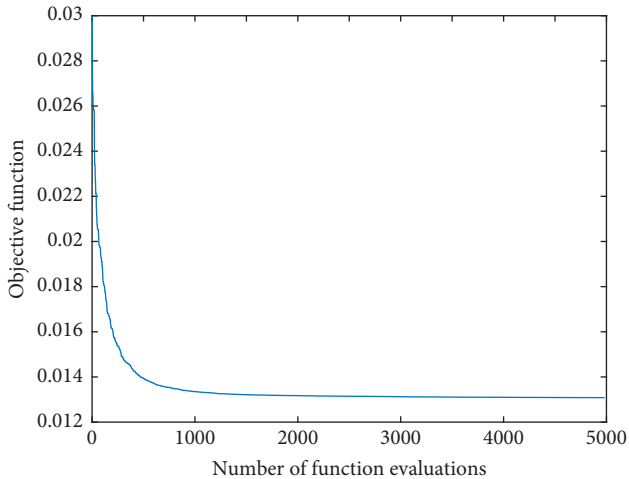


Figure 4: Convergence curve of SAR for the I-beam design problem.

5.5.3. Spatial 25-Bar Truss Structure Design. SAR was compared with different methods including the artificial bee colony algorithm with an adaptive penalty function approach (ABC-AP) [67], self-adaptive harmony search algorithm (SAHS) [68], teaching-learning-based optimization algorithm (TLBO) [69], multistage particle swarm optimization (MSPSO) [70], hybrid particle swallow swarm optimization (HPSSO) [71], water evaporation optimization (WEO) [72], and culture algorithm (CA) [73]. Table 16 presents the optimization results of these methods.

Regarding the data in this table, SAR found the best design (545.0365 lb) and required less structural analyses than the other algorithms. The average and standard deviation of SAR are 545.0391 lb and 0.0064 lb, respectively, and they are significantly better than those of the other algorithms. These results clearly indicate that SAR outperforms the other algorithms. Figure 6 shows the convergence curve of SAR for the spatial 25-bar truss design problem.

5.6. Analysis of SAR Parameters. In this section, the effect of SAR parameters on solving the benchmark problems was investigated. Two classic ($f13$ and $f18$) and four modern ($C3$, $C10$, $C22$, and $C28$) benchmark functions were utilized for this test. $f13$ and $C3$ are unimodal functions, and $f18$, $C10$, $C22$, and $C28$ are multimodal functions. For classic and modern functions, NFEs were set to $2 \times 10^4 \times \mathrm{D}$ and 10^5, respectively. The population of SAR was equal to 20, and it

Table 15: Comparison results for the cantilever beam design.

Variables	MMA	GCA(I)	GCA(II)	SOS	CS	FPA	MFA	SAR
$x1$	6.01	6.01	6.01	6.01878	6.0089	6.0202	6.01422	6.016081
$x2$	5.3	5.3	5.3	5.30344	5.3049	5.3082	5.3122	5.309224
$x3$	4.49	4.49	4.49	4.49587	4.5023	4.5042	4.48929	4.494135
$x4$	3.49	3.49	3.49	3.49896	3.5077	3.4856	3.50375	3.501578
$x5$	2.15	2.15	2.15	2.15564	2.1504	2.1557	2.15422	2.152641
$f_{\min}$	1.34	1.34	1.34	1.33996	1.33999	1.33997	1.339957	**1.3399563**
Average	NA	NA	NA	1.33997	NA	NA	NA	**1.3399564**
Std.	NA	NA	NA	$1.10E-05$	NA	NA	NA	**2.73E−08**
NFE	NA	NA	NA	15000	250000	15000	15000	10000

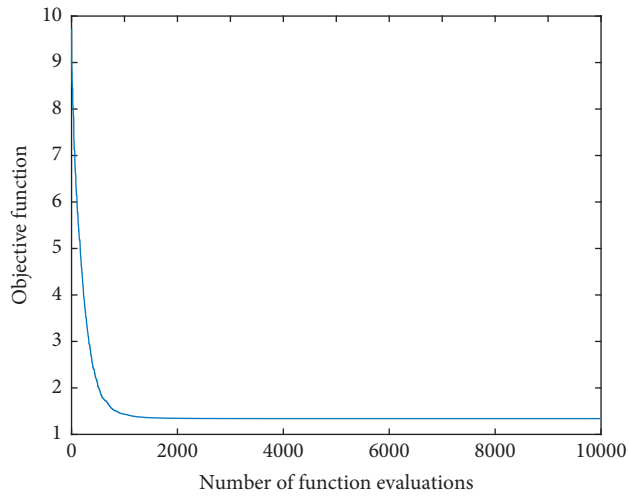


Figure 5: Convergence curve of SAR for the cantilever beam design problem.

Table 16: Comparison results for the spatial 25-bar truss structure.

Element groups	ABC-AP	SAHS	TLBO	MSPSO	HPSSO	WEO	CA	SAR
1	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
2	1.979	2.074	2.0712	1.9848	1.9907	1.9814	2.02064	2.042677
3	3.003	2.961	2.957	2.9956	2.9881	3.0023	3.01733	3.002584
4	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
5	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
6	0.69	0.691	0.6891	0.6852	0.6824	0.6827	0.69383	0.683408
7	1.679	1.617	1.6209	1.6778	1.6764	1.6778	1.63422	1.623117
8	2.652	2.674	2.6768	2.6599	2.6656	2.6612	2.65277	2.671827
$f_{\min}$	545.193	545.12	545.09	545.16	545.164	545.166	545.05	**545.0365**
Average	NA	545.94	545.41	546.03	545.556	545.226	545.93	**545.0391**
Std.	NA	0.91	0.42	0.8	0.432	0.083	1.55	**0.0064**
NFE	300000	9051	15318	10800	13326	19750	9380	9000

was independently executed 20 times for each configuration, and errors less than 10^{-8} were considered zero. The means of errors obtained by SAR for different values of the SE are reported in Table 17. To evaluate the effect of the SE on the performance of SAR, the MU was set to a big value because the unimproved clues will not be left for this value of MU. Hence, the effect of the MU was removed. According to the data in Table 17, the optimum value of the SE depends on the nature of the problem. The high values of the SE (more than 0.8) lead to unsatisfactory performance of SAR. Furthermore, SE values between 0.6 and 0.8 are suitable for unimodal functions and SE values lower than 0.1 are suitable for multimodal functions. Solving multimodal functions generally is more difficult than unimodal functions. Therefore, the SE was set to 0.05 for all the studied problems in this paper.

Table 18 shows the means of errors obtained by SAR with fixed SE = 0.05 for different values of the MU. From this table, it can be seen that the lower value of this parameter decreases the convergence rate of the algorithm and increases the chance of avoiding local minima. According to the results, the best value of the MU for the studied problems

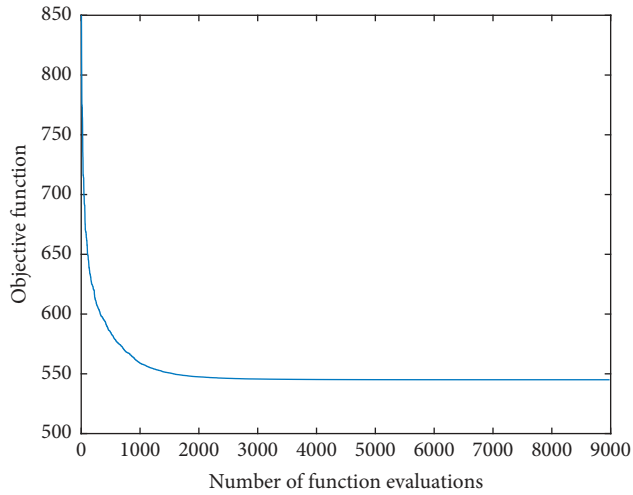


Figure 6: Convergence curve of SAR for the spatial 25-bar truss structure problem.

Table 17: Effects of the SE on the performance of SAR with fixed MU = 1000 × D.

SE	$f13$	$f18$	$C3$	$C10$	$C22$	$C28$
0	$7.97E-01$	$0.00E+00$	$4.28E+00$	$2.67E-02$	$1.03E+01$	$2.30E+02$
0.05	$3.99E-01$	$9.95E-02$	$1.42E+00$	$2.47E-02$	$1.12E+01$	$2.40E+02$
0.1	$5.98E-01$	$5.47E-01$	$1.56E+00$	$2.79E-02$	$1.27E+01$	$2.20E+02$
0.2	$7.97E-01$	$2.49E+00$	$1.71E+00$	$2.45E-02$	$1.55E+01$	$2.50E+02$
0.3	$1.40E+00$	$5.02E+00$	$2.37E+00$	$2.50E-02$	$2.39E+01$	$2.60E+02$
0.4	$1.20E+00$	$9.95E+00$	$1.24E+00$	$2.66E-02$	$4.03E+01$	$2.60E+02$
0.5	$1.79E+00$	$1.62E+01$	$3.30E-01$	$2.68E-02$	$5.57E+01$	$2.80E+02$
0.6	$5.98E-01$	$2.14E+01$	$1.58E+00$	$4.15E-02$	$8.59E+01$	$2.93E+02$
0.7	$9.97E-01$	$3.28E+01$	$9.57E-01$	$6.73E-02$	$1.15E+02$	$3.13E+02$
0.8	$1.20E+00$	$4.41E+01$	$2.69E-03$	$7.34E-02$	$1.99E+02$	$3.19E+02$
0.9	$1.81E+00$	$6.90E+01$	$9.33E-01$	$2.14E-01$	$3.50E+02$	$3.91E+02$
1	$3.87E+06$	$1.76E+02$	$7.43E+08$	$5.75E+01$	$1.15E+03$	$7.47E+02$

Table 18: Effects of the MU on the performance of SAR with fixed SE = 0.05.

MU	$f13$	$f18$	$C3$	$C10$	$C22$	$C28$
$5 \times D$	$0.00E+00$	$0.00E+00$	$1.33E+05$	$3.24E-01$	$1.12E+02$	$1.80E+02$
$10 \times D$	$0.00E+00$	$0.00E+00$	$7.61E+02$	$7.78E-02$	$2.25E+01$	$1.60E+02$
$30 \times D$	$0.00E+00$	$0.00E+00$	$3.1E+00$	$3.56E-02$	$1.09E+01$	$2.00E+02$
$50 \times D$	$0.00E+00$	$0.00E+00$	$2.66E+00$	$3.43E-02$	$1.09E+01$	$2.00E+02$
$70 \times D$	$0.00E+00$	$0.00E+00$	$1.59E+00$	$3.09E-02$	$9.62E+00$	$2.10E+02$
$90 \times D$	$1.99E-01$	$0.00E+00$	$1.72E+00$	$3.23E-02$	$1.15E+01$	$2.20E+02$
$150 \times D$	$1.99E-01$	$0.00E+00$	$1.32E+00$	$2.83E-02$	$1.16E+01$	$2.50E+02$
$250 \times D$	$1.99E-01$	$0.00E+00$	$1.75E+00$	$2.34E-02$	$1.07E+01$	$2.40E+02$
$1000 \times D$	$3.99E-01$	$9.95E-02$	$1.42E+00$	$2.47E-02$	$1.12E+01$	$2.40E+02$

is $70 \times D$. However, SAR is not very sensitive to this parameter. The results indicate the effect of the SE on the performance of SAR is more than the effect of the MU. Thus, the SE is the key parameter of the proposed algorithm.

6. Conclusion

A new metaheuristic optimization algorithm called the search and rescue optimization algorithm (SAR) was introduced here for solving single-objective optimization problems. SAR was inspired by the explorations carried out by humans during search and rescue operations. The proposed algorithm consists of two phases including the social phase and the individual phase, and the implementation of it is relatively simple. The results of the tests done in this paper have shown that combining these two phases along with the use of memory leads to a balance between exploration and exploitation processes in SAR.

The performance of SAR was compared with that of twelve different optimization algorithms through fifty-five

continuous benchmark problems including a set of 27 classic benchmark problems and a set of 28 modern CEC 2013 benchmark functions. The compared algorithms included recent variants of DE (SaDE and RJADE), GA (AMopGA), PSO (SPSO and f_k-PSO), and CMA-ES (CMA-RIS) algorithms and some recent metaheuristic algorithms (ABC, GSA, TLBO, GWO, and YYPO). The Wilcoxon signed-rank test was used for some of the comparisons, and the convergence behavior of SAR was investigated. The statistical results indicated SAR is suitable for global optimization and highly competitive with the other algorithms. The proposed algorithm performed better than most of the compared algorithms in terms of finding the global optimum and convergence rate in many cases. Besides, to verify the application of the proposed algorithm on real-world optimization problems, SAR was tested on three engineering design problems including the I-beam design, the cantilever beam, and the spatial 25-bar truss structure design. The obtained results revealed that the proposed algorithm can find more accurate solutions with fewer function evaluations in comparison with the other existing algorithms. In future works, the performance of SAR on other types of optimization problems such as combinatorial and large-scale optimization problems will be investigated.

Data Availability

The data supporting this study are from previously reported studies and datasets, which have been properly cited in this paper. All problems were completely defined in this paper, except CEC 2013 instances. The codes of CEC 2013 benchmark instances and the algorithms implemented for this study were downloaded from the following links: The CEC 2013 benchmark data used to support the findings of this study have been deposited in the CEC2013matlab repository at http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared%20Documents/CEC2013/ce_c13matlab.zip. The MATLAB source code of SAR used to support the findings of this study are available from Amir Shabani upon request (email: amir48ash@gmail.com). The algorithms used to compare the proposed algorithm are available from the following links: ABC: http://mf.erciyes.edu.tr/abc/; DE: https://ch.mathworks.com/matlabcentral/fileexchange/52897-differentialevolution-de; GSA: http://www.mathworks.com/matlabcentral/fileexchange/27756-gravitationalsearch-algorithm-gsa; and TLBO: https://ch.mathworks.com/matlabcentral/fileexchange/52863-teaching-learningbased-optimization-tlbo.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was partially supported by the Spanish Research Project (nos. TIN2016-80856-R and TIN2015-65515-C4-1-R).

References

[1] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "A survey on metaheuristics for stochastic combinatorial optimization," *Natural Computing*, vol. 8, no. 2, pp. 239–287, 2009.

[2] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–72, 1992.

[3] M. Kumar, M. Husian, N. Upreti, and D. Gupta, "Genetic algorithm: review and application," *International Journal of Information Technology and Knowledge Management*, vol. 2, no. 2, pp. 451–454, 2010.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, Perth, WA, Australia, December 1995.

[5] K. L. Du and M. Swamy, *Particle Swarm Optimization. Search and Optimization by Metaheuristics*, Springer, Berlin, Germany, 2016.

[6] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[7] K. L. Du and M. Swamy, *Ant Colony Optimization. Search and Optimization by Metaheuristics*, Springer, Berlin, Germany, 2016.

[8] 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.

[9] D. Manjarres, I. Landa-Torres, S. Gil-Lopez et al., "A survey on applications of the harmony search algorithm," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 8, pp. 1818–1831, 2013.

[10] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (abc) algorithm and applications," *Artificial Intelligence Review*, vol. 42, no. 1, pp. 21–57, 2014.

[11] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Kayseri, Turkey, 2005.

[12] X. S. Yang and S. Deb, "Cuckoo search via lévy flights," in *Nature & Biologically Inspired Computing, 2009 (NaBIC 2009), World Congress on 2009*, IEEE, Piscataway, NJ, USA, 2009.

[13] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 4661–4667, Singapore, September 2007.

[14] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems," *Computer-Aided Design*, vol. 43, no. 3, pp. 303–315, 2011.

[15] C. Zhang, Q. Lin, L. Gao, and X. Li, "Backtracking search algorithm with three constraint handling methods for constrained optimization problems," *Expert Systems with Applications*, vol. 42, no. 21, pp. 7831–7845, 2015.

[16] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.

[17] V. Punnathanam and P. Kotecha, "Yin-yang-pair optimization: a novel lightweight optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 54, pp. 62–79, 2016.

[18] M. Jain, V. Singh, and A. Rani, "A novel nature-inspired algorithm for optimization: squirrel search algorithm," *Swarm and Evolutionary Computation*, vol. 44, 2019.

[19] C. Zhao, C. Wu, J. Chai et al., "Decomposition-based multi-objective firefly algorithm for rfid network planning with uncertainty," *Applied Soft Computing*, vol. 55, pp. 549–564, 2017.

[20] C. Zhao, C. Wu, X. Wang et al., "Maximizing lifetime of a wireless sensor network via joint optimizing sink placement and sensor-to-sink routing," *Applied Mathematical Modelling*, vol. 49, pp. 319–337, 2017.

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

[22] T. Joyce and J. M. Herrmann, "A review of no free lunch theorems, and their implications for metaheuristic optimisation," in *Nature-inspired Algorithms and Applied Optimization*, pp. 27–51, Springer, Berlin, Germany, 2018.

[23] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[24] H. Garg, "An efficient biogeography based optimization algorithm for solving reliability optimization problems," *Swarm and Evolutionary Computation*, vol. 24, pp. 1–10, 2015.

[25] R. Storn and K. Price, "Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[26] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution—an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[27] I. D. Couzin, J. Krause, N. R. Franks, and S. A. Levin, "Effective leadership and decision-making in animal groups on the move," *Nature*, vol. 433, no. 7025, pp. 513–516, 2005.

[28] A. H. Gandomi, A. H. Alavi, and K. herd, "Krill Herd: a new bio-inspired optimization algorithm," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 12, pp. 4831–4845, 2012.

[29] A. Kaveh and P. Hosseini, "A simplified dolphin echolocation optimization method for optimum design of trusses," *International Journal of Optimization in Civil Engineering*, vol. 4, no. 3, pp. 381–397, 2014.

[30] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

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

[32] A. Kaveh and V. R. Mahdavi, "Colliding bodies optimization: a novel meta-heuristic method," *Computers & Structures*, vol. 139, pp. 18–27, 2014.

[33] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "Gsa: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[34] E. Rashedi, E. Rashedi, and H. Nezamabadi-pour, "A comprehensive survey on gravitational search algorithm," *Swarm and Evolutionary Computation*, vol. 41, 2018.

[35] S. A. Gharebaghi, A. Kaveh, and M. A. Asl, "A new meta-heuristic optimization algorithm using star graph," *Smart Structures and Systems*, vol. 20, no. 1, pp. 99–114, 2017.

[36] Y.-J. Zheng, "Water wave optimization: a new nature-inspired metaheuristic," *Computers & Operations Research*, vol. 55, pp. 1–11, 2015.

[37] A. Kaveh and M. Khayatazad, "A new meta-heuristic method: ray optimization," *Computers & Structures*, vol. 112-113, pp. 283–294, 2012.

[38] F. Glover, "Tabu search-part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[39] H.-P. Chiang, Y.-H. Chou, C.-H. Chiu, S.-Y. Kuo, and Y.-M. Huang, "A quantum-inspired tabu search algorithm for solving combinatorial optimization problems," *Soft Computing*, vol. 18, no. 9, pp. 1771–1781, 2014.

[40] S. J. Mousavirad and H. Ebrahimpour-Komleh, "Human mental search: a new population-based metaheuristic optimization algorithm," *Applied Intelligence*, vol. 47, no. 3, pp. 850–887, 2017.

[41] R. Rubio-Marín, *Human Rights and Immigration*, Oxford University Press, Oxford, UK, 2014.

[42] R. H. Major, *Early Voyages to Terra Australis, Now Called Australia*, РиПол Классик, Moscow, Russia, 1859.

[43] ASTM F1848, *Standard Classification for Search and Rescue Dog Crew/teams*, ASTM, West Conshohocken, PA, USA, 2005.

[44] ASTM F1847, *Standard Guide for Demonstrating Minimum Skills of Search and Rescue Dogs*, ASTM, West Conshohocken, PA, USA, 2012.

[45] K. E. Management, "Sar field search methods: search techniques used by trained teams in the field, kentucky emergency management," 2019.

[46] 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.

[47] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems," *Information Sciences*, vol. 183, no. 1, pp. 1–15, 2012.

[48] J. G. Digalakis and K. G. Margaritis, "On benchmarking functions for genetic algorithms," *International Journal of Computer Mathematics*, vol. 77, no. 4, pp. 481–506, 2001.

[49] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[50] M. Molga and C. Smutnicki, *Test Functions for Optimization Needs*, 2005, https://www.vafaeijahan.com/en/wp-content/uploads/2012/02/Test-functions-for-optimization-needs.pdf.

[51] J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Díaz, *Problem Definitions and Evaluation Criteria for the Cec 2013 Special Session on Real-Parameter Optimization*, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China, 2012.

[52] F. Caraffini, G. Iacca, F. Neri, L. Picinali, and E. Mininno, "A cma-es super-fit scheme for the re-sampled inheritance search," in *Evolutionary Computation (CEC), 2013 IEEE Congress on, 2013*, pp. 1123–1130, IEEE, Piscataway, NJ, USA, 2013.

[53] T. Y. Lim, M. A. Al-Betar, and A. T. Khader, "Adaptive pair bonds in genetic algorithm: an application to real-parameter optimization," *Applied Mathematics and Computation*, vol. 252, pp. 503–519, 2015.

[54] R. A. Khanum, N. Tairan, M. A. Jan, W. K. Mashwani, and A. Salhi, "Reflected adaptive differential evolution with two external archives for large-scale global optimization," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, pp. 675–683, 2016.

[55] A. K. Qin, X. Li, H. Pan, and S. Xia, "Investigation of self-adaptive differential evolution on the cec-2013 real-parameter single-objective optimization testbed," in *Evolutionary*

Computation (CEC), 2013 IEEE Congress on, 2013, pp. 1107–1114, IEEE, Piscataway, NJ, USA, 2013.

[56] F. V. Nepomuceno and A. P. Engelbrecht, "A self-adaptive heterogeneous pso for real-parameter optimization," in *Evolutionary Computation (CEC), 2013 IEEE Congress on, 2013*, pp. 361–368, IEEE, Piscataway, NJ, USA, 2013.

[57] M. Zambrano-Bigiarini, M. Clerc, and R. Rojas, "Standard particle swarm optimisation 2011 at cec-2013: a baseline for future pso improvements," in *Evolutionary Computation (CEC), 2013 IEEE Congress on, 2013*, pp. 2337–2344, IEEE, Piscataway, NJ, USA, 2013.

[58] S. Gold and S. Krishnamurty, "Trade-offs in robust engineering design," in *Proceedings of the ASME International Design Engineering Technical Conference*, Sacramento, CA, USA, 1997.

[59] C. Fleury and V. Braibant, "Structural optimization: a new dual method using mixed variables," *International Journal for Numerical Methods in Engineering*, vol. 23, no. 3, pp. 409–428, 1986.

[60] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[61] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering with Computers*, vol. 29, no. 1, pp. 17–35, 2013.

[62] G. G. Wang, "Adaptive response surface method using inherited Latin hypercube design points," *Journal of Mechanical Design*, vol. 125, no. 2, pp. 210–220, 2003.

[63] M.-Y. Cheng and D. Prayogo, "Symbiotic organisms search: a new metaheuristic optimization algorithm," *Computers & Structures*, vol. 139, pp. 98–112, 2014.

[64] H. Chickermane and H. C. Gea, "Structural optimization using a new local approximation method," *International Journal for Numerical Methods in Engineering*, vol. 39, no. 5, pp. 829–846, 1996.

[65] S. M. Nigdeli, G. Bekdaş, and X. S. Yang, *Application of the Flower Pollination Algorithm in Structural engineering. Metaheuristics and Optimization in Civil Engineering*, Springer, Berlin, Germany, 2016.

[66] J.-S. Chou and N.-T. Ngo, "Modified firefly algorithm for multidimensional optimization in structural design problems," *Structural and Multidisciplinary Optimization*, vol. 55, no. 6, pp. 2013–2028, 2017.

[67] M. Sonmez, "Artificial bee colony algorithm for optimization of truss structures," *Applied Soft Computing*, vol. 11, no. 2, pp. 2406–2418, 2011.

[68] S. O. Degertekin, "Improved harmony search algorithms for sizing optimization of truss structures," *Computers & Structures*, vol. 92-93, pp. 229–241, 2012.

[69] S. O. Degertekin and M. S. Hayalioglu, "Sizing truss structures using teaching-learning-based optimization," *Computers & Structures*, vol. 119, pp. 177–188, 2013.

[70] S. Talatahari, M. Kheirollahi, C. Farahmandpour, and A. H. Gandomi, "A multi-stage particle swarm for optimum design of truss structures," *Neural Computing and Applications*, vol. 23, no. 5, pp. 1297–1309, 2013.

[71] A. Kaveh, T. Bakhshpoori, and E. Afshari, "An efficient hybrid particle swarm and swallow swarm optimization algorithm," *Computers & Structures*, vol. 143, pp. 40–59, 2014.

[72] A. Kaveh and T. Bakhshpoori, "A new metaheuristic for continuous structural optimization: water evaporation optimization," *Structural and Multidisciplinary Optimization*, vol. 54, no. 1, pp. 23–43, 2016.

[73] S. Jalili and Y. Hosseinzadeh, "A cultural algorithm for optimal design of truss structures," *Latin American Journal of Solids and Structures*, vol. 12, no. 9, pp. 1721–1747, 2015.