Expert Systems with Applications 161 (2020) 113698

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Search and rescue optimization algorithm: A new optimization method for solving constrained engineering optimization problems

Amir Shabani [a], Behrouz Asgarian [a,*], Miguel Salido [b], Saeed Asil Gharebaghi [a]

[a] Faculty of Civil Engineering, K.N. Toosi University of Technology, Tehran, Iran
[b] Instituto de Automatica e Informatica Industrial, Universitat Politecnica de Valencia, Valencia, Spain

ARTICLE INFO

Article history:
Received 26 August 2019
Revised 10 June 2020
Accepted 25 June 2020
Available online 4 July 2020

Keywords:
Constrained optimization
Engineering optimization problems
Search and rescue optimization algorithm
Metaheuristic algorithms

ABSTRACT

A new optimization method namely the Search and Rescue optimization algorithm (SAR) is presented here to solve constrained engineering optimization problems. This metaheuristic algorithm imitates the explorations behavior of humans during search and rescue operations. The ε-constrained method is utilized as a constraint-handling technique. Besides, a restart strategy is proposed to avoid local infeasible minima in some complex constrained optimization problems. SAR is applied to solve 18 benchmark constraint functions presented in CEC 2010, 13 benchmark constraint functions, and 7 constrained engineering design problems reported in the specialized literature. The performance of SAR is compared with some state-of-the-art optimization algorithms. According to the statistical comparison results, the performance of SAR is better or highly competitive against the compared algorithms on most of the studied problems.

1. Introduction

Optimization is the method of finding an optimal solution from all available feasible solutions. Although many optimization methods have been developed so far, regarding the No Free Lunch Theorem (NFL) (Wolpert & Macready, 1997), there is no optimization algorithm which works well on all optimization problems. Therefore, an optimization algorithm might be efficient for just a series of optimization problems. There are many applications for optimization and many kinds of optimization problems (Deb, 2012; Yang, 2010). For instance, many optimization algorithms have been proposed for machine learning so far (Young, Hinkle, Kannan, & Ramanathan, 2020; Zhang, Ding, Zhang, & Xue, 2018). Optimization algorithms have been utilized to minimize the training error of deep learning and they directly affect the efficiency of deep learning's training (Zhang & Ding, 2017; Zhang, Ding, Zhang, & Xue, 2017). Another kind of optimization problems are engineering design problems. In engineering design, designers are looking for optimum design parameters to minimize materials usage, financial resources time or cost and maximize efficiency, performance, and quality or lifetime service (Deb, 2012; Yang, 2010). Classical optimization methods such as mathematical programming are regularly efficient for simple optimization problems and some of these methods use gradient information to find an optimum solution, but real-world engineering optimization problems are often nonlinear, non-differentiable, complex and multimodal. These problems are relatively difficult to solve using classical optimization methods. Hence, a global optimization algorithm, which does not need gradient information, is required to solve those (Cheng & Prayogo, 2017).

In recent decades, many methods have been investigated and applied to solve engineering optimization problems. Among different methods, metaheuristic algorithms have shown to maintain acceptable performance (Osman & Laporte, 1996). Unlike classical optimization methods, metaheuristic algorithms do not need gradient information and can escape from local optima; therefore, they are suitable for solving engineering optimization problems (Torres-Jiménez & Pavón, 2014). They can find an optimal solution regardless of the physical nature of the problem. Most of them are inspired by physical or natural phenomena. For example, particle swarm optimization (PSO) (Du & Swamy, 2016; Kennedy and Eberhart, 1995) is inspired by the social foraging behavior of bird flocking or fish schooling. In the past few years, many metaheuristic algorithms have been proposed to solve engineering optimization problems such as bat algorithm (BAT) (Yang & Hossein Gandomi, 2012), cuckoo search (CS) (Gandomi, Yang, & Alavi, 2013), whale optimization algorithm (WOA) (Mirjalili & Lewis, 2016), firefly algorithm (FA) (Gandomi, Yang, & Alavi, 2011), artificial electric field algorithm (AEFA) (Yadav & Kumar, 2020), star graph (Gharebaghi, Kaveh, & Asl, 2017), krill herd (KH) (Gandomi

* Corresponding author.
E-mail addresses: amir48ash@gmail.com (A. Shabani), Asgarian@kntu.ac.ir (B. Asgarian), msalido@dsic.upv.es (M. Salido), asil@kntu.ac.ir (S. Asil Gharebaghi).

https://doi.org/10.1016/j.eswa.2020.113698
0957-4174/

& Alavi, 2012), an improved firefly algorithm (AD-IFA) (Wu, et al., 2020), crow search algorithm (CSA) (Askarzadeh, 2016) and differential big bang-big crunch (DDB-BC) (Prayogo, Cheng, Wu, Herdany, & Prayogo, 2018) algorithms and many of them were applied to solve constrained engineering optimization problems.

Most of engineering optimization problems are constrained. The general form of a single objective constrained optimization problem can be formulated as a minimization problem, as follows:

$$\begin{aligned} &\text{minimize } f\left(\vec{x}\right), \vec{x} = (x_1, \cdots, x_D), x_i^{min} \le x_i \le x_i^{max} \\ &\text{subject to: } g_j\left(\vec{x}\right) \le 0, j = 1, \cdots, n \\ &h_j\left(\vec{x}\right) = 0, j = n+1, \cdots, m \end{aligned} \tag{1}$$

where $\vec{x}$ is a solution and $f\left(\vec{x}\right)$ is the objective function for this solution. x_i^{min} and x_i^{max} are the values of the maximum and minimum threshold for the i^{th} dimension, respectively. $g_j\left(\vec{x}\right)$ is the j^{th} inequality constraint, and $h_j\left(\vec{x}\right)$ is the $(j\text{-}n)^{\text{th}}$ equality constraint. The equality constraints are converted into inequality constraints by a tolerance value δ, as follows:

$$\left|h_j\left(\vec{x}\right)\right| - \delta \le 0 \tag{2}$$

δ is a small positive constant. Finally, the constraint violation is given by

$$\begin{aligned} &G\left(\vec{x}\right) = \sum_{j=1}^{m} G_j\left(\vec{x}\right) \\ &G_j\left(\vec{x}\right) = \begin{cases} \max\left(0, g_j\left(\vec{x}\right)\right) & 1 \le j \le \text{n} \\ \max\left(0, \left|h_j\left(\vec{x}\right)\right| - \delta\right) & \text{n}+1 \le j \le \text{m} \end{cases} \end{aligned} \tag{3}$$

where $G_j\left(\vec{x}\right)$ is the constraint violation of the j^{th} constraint.

In this paper, a new metaheuristic optimization method called Search and Rescue optimization algorithm (SAR) is proposed to solve constrained engineering optimization problems. It is inspired by the explorations conducted during search and rescue operations. The preliminary investigated indicated the efficiency of this algorithm for solving unconstrained and discrete optimization problems (Shabani, Asgarian, Gharebaghi, Salido, & Giret, 2019b; Shabani, Asgarian, & Salido, 2019a). In this study, the performance of SAR is validated against several constrained optimization problems and the results are compared with some state-of-the-art optimization algorithms.

The rest of the paper is organized as follows: Following the introduction in Section 1, Section 2 introduces search and rescue operation. Then, the SAR algorithm is described in Section 3. Section 4 presents the experimental results and discussion. Finally, Section 5 gives the concluding remarks.

2. Search and rescue operation

In nature, many creatures live socially and search for a variety of goals such as hunting and finding food sources as groups. They use different strategies for searching (Couzin, Krause, Franks, & Levin, 2005). Similarly, human beings search for different purposes as groups. For instance, they search for hunting, finding food sources or lost people. One type of group explorations is the 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 (Rubio-Marín, 2014). Search and rescue operations are sometimes carried out to find specific people who are lost. Humans search in groups, which are consisted of group members, and each of these searching groups organizes their practices to be prepared for their respective operations (ASTM, 2005). Humans (group members) can identify the clues and traces of lost people based on the received training. The found clues are of different significance and provide different information about lost individuals. For instance, 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 these information of found clues via communication equipment to other members. Finally, they search based on the significance degree of these clues and information that can be obtained from them. The group members generally search around the found clues or connect them together and search in those directions (Kentucky Emergency Management, 2020; OSARVA Senior Search and Rescue Trainers, 2016). Thus, the procedure of human searches, in search and rescue operations, can be divided into two phases: *individual* and *social*. In the individual search phase, the searching is done regardless of the position and the significance of clues found by others. In the social phase, the group members are searching based on the position of found clues and their significance in areas that are likely to get more significant clues. There are two types of clues:

1. Hold clue: one group member is present there and searches around it.
2. Abandoned clue: The group member who found the clue has left it to find more significant clues, but the information of this clue is available for others.

The two phases of human search in search and rescue operations are shown in Fig. 1. According to this figure, the locations of a group member and a clue are indicated by points **A** and **B**, respectively. Path1, Path2, and Path3 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 more significant clues increases the probability of finding the lost person, the area that has more significant clues in the direction of **AB** will be selected. In other words, if there are more significant clues in area 1 compared to area 2, area 1 is chosen. Otherwise, area 2 is selected to be searched. In the sample case depicted in Fig. 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 more significant 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, assumingly 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 more significant clues than in the previous location (position **A**), the new location becomes the new position of the member of the search group.

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 this model, the humans' position corresponds to the solution of the optimization problem and the significance of clue found in this position indicates the fitness for this solution. A better solution represents a more significant clue and vice versa.

3.1. Clues

During the search operation, group members gather clues information. To find more significant clues, some clues are left by group members, but their information is used to improve the searching operation. In the model we proposed, the left clues positions are

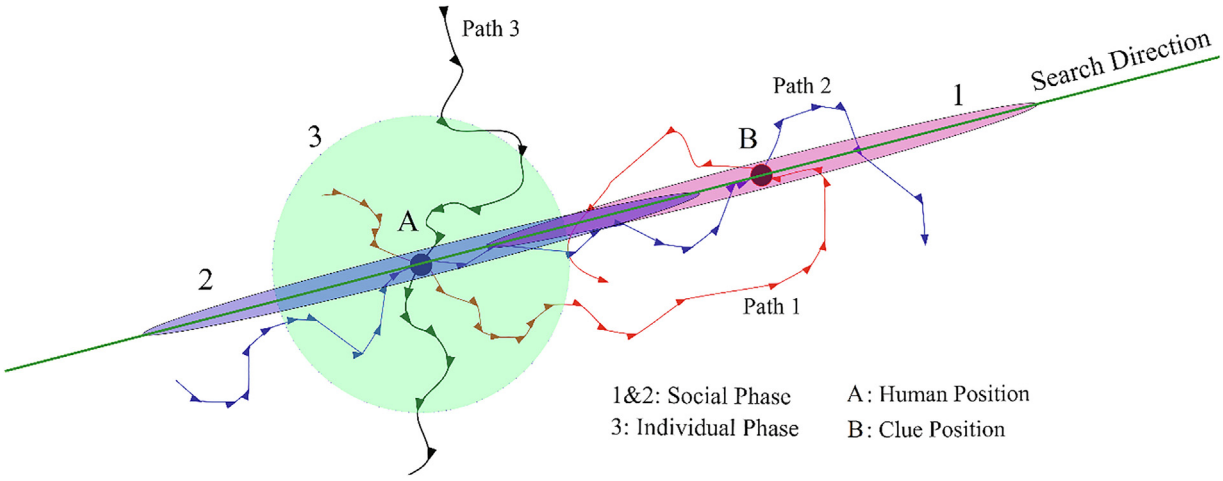

Fig. 1. Two types of human search in search and rescue operations.

stored in a memory matrix (matrix M), whereas the humans' positions are stored in a position matrix (matrix X). The dimensions of matrix M are equal to matrix X. They are $N \times D$ matrices where D is the dimension of the problem and N indicates the number of group members. Clues matrix (matrix C) is a matrix containing the positions of found clues. This matrix consists of two matrices X and M. Eq. (4) shows how to create C. All new solutions in social and individual phases are created based on the clues matrix and it is an essential part of SAR. These matrices (X, M, and C) are updated in each human searches phases.

$$c = \begin{bmatrix} X \\ M \end{bmatrix} = \begin{bmatrix} X_{11} & \cdots & X1D \\ \vdots & \ddots & \vdots \\ X_{N1} & \cdots & X_{ND} \\ M_{11} & \cdots & M_{1D} \\ \vdots & \ddots & \vdots \\ M_{N1} & \cdots & M_{ND} \end{bmatrix} \tag{4}$$

where **M** and **X** are memory and humans' positions matrices, respectively and X_{N1} is the position of the 1st dimension for the N^{th} human. Moreover, M_{1D} is the position of the D^{th} dimension for the 1st memory. In the following section, the two phases of humans' search, including the "social phase" and the "individual phase", are modelled.

3.2. Social phase

According to the previous explanations and also taking into account a random clue among found clues, Eq. (5) is used to obtain the search direction.

$$SD_i = (X_i - C_k), k \neq i \tag{5}$$

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 for the i^{th} human, respectively. k is a random integer number ranged between 1 and *2 N*. For $i = k$, C_i will be equal to X_i. So, k is chosen in such a way that $k \neq i$.

The group members normally try to avoid searching for a location many times. So, the search should be done in a manner that the 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 Eq. (5). To apply this constraint, the binomial crossover operator has been used. Also as explained in the previous section, if the considered clue is more significant than the clue related to the current position (Objective function value for solution B is greater than objective function value for solution A in Fig. 1), an area around SD_i direction and around the position of that clue is searched (area 1 in Fig. 1); otherwise, the search will be continued around the current position along with SD_i direction (area 2 in Fig. 1). So, Eq. (6) is used for the social phase:

$$X'_{i,j} = \begin{cases} \begin{cases} C_{k,j} + r1 \times (X_{i,j} - C_{k,j}) & \text{if } f(C_k) > f(X_i) \\ X_{i,j} + r1 \times (X_{i,j} - C_{k,j}) & \text{otherwise} \end{cases} & \text{if } r2 < SE \text{ or } j = j_{rand}, j = 1, \cdots, D \\ X_{i,j} & \text{otherwise} \end{cases} \tag{6}$$

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} clue. $f(C_k)$ and $f(X_i)$ are the objective function values for the solution $\mathbf{C_k}$ and $\mathbf{X_i}$, 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]. *r2* is different for each dimension, but *r1* is fixed for all dimensions. j_{rand} is a random integer number ranged between 1 and D which ensures that at least one dimension of $X'_{i,j}$ is different from $X_{i,j}$. *SE* is an algorithm parameter ranged between 0 and 1. The new position of the i^{th} human in all dimensions is obtained by Eq. (6).

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 employed for this phase. The new position of the i^{th} human is obtained by Eq. (7).

$$X'_i = X_i + r3 \times (C_k - C_m), i \neq k \neq m \tag{7}$$

where k and m are random integer numbers ranged between 1 and *2 N*. To prevent movement along 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 ranged between 0 and 1.

3.4. Boundary control

The solutions obtained by the social and individual phases should be located in the solution space, and if they are out of it, they should be modified. To this end, Eq.8 is used to modify the new position of the i^{th} human.

$$X'_{i,j} = \begin{cases} \left(X_{i,j} + X_j^{max}\right)/2 & if\ X'_{i,j} > X_j^{max} \\ \left(X_{i,j} + X_j^{min}\right)/2 & if\ X'_{i,j} < X_j^{min} \end{cases}, j = 1, \cdots, D \quad (8)$$

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. Update information and positions

In each iteration, the group members will search according to these two phases and after each phase, if the value of the objective function in position X'$_i$ ($f\left(X'_i\right)$) 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 Eq. (9) and this position will be accepted as a new position using Eq. (10). Otherwise, this position is left and the memory is not updated.

$$M_n = \begin{cases} X_i & if\ f\left(X'_i\right) > f(X_i) \\ M_n & otherwise \end{cases} \quad (9)$$

$$X_i = \begin{cases} X'_i & if\ f\left(X'_i\right) > f(X_i) \\ X_i & otherwise \end{cases} \quad (10)$$

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

3.6. Abandoning clues

In search and rescue operations, time is a very important factor since 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 group member cannot find more significant clues after a certain number of searches around his/her current position, s/he 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 group member. Whenever a human finds more significant clues in the first or second phase of the search, the USN is set to 0 for that human; otherwise, it will be increased by 1 point as presented in Eq. (11).

$$USN_i = \begin{cases} \mathrm{USN}_i + 1 & if\ f\left(X'_i\right) < f(X_i) \\ 0 & otherwise \end{cases} \quad (11)$$

Where USN_i indicates the number of times the i^{th} human has not been able to find more significant clues. When the USN for a human is greater than MU, s/he goes to another position in the search space. In order to solve constrained optimization problems, for a feasible solution, if $USN > MU$, the current solution is replaced with a random solution in the search space using Eq. (12). Also, for an infeasible solution, if $USN > MU$, the solution in the memory matrix with a minimum degree of constraint violation is selected and the current solution is substituted with this solution and the current solution takes its place in the memory matrix.

$$X_{i,j} = X_j^{min} + r4 \times \left(X_j^{max} - X_j^{min}\right), j = 1, \cdots, D \quad (12)$$

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

3.7. Constraint-handling technique

There are several constraint-handling methods such as stochastic ranking, penalty functions approach and ε-constrained method. The penalty functions methods are very popular for solving constrained optimization problems, however they are sensitive to penalty factors. The ε-constrained method is one of the popular constraint-handling methods that is used in this study. According to this method, for a maximization problem, a solution is considered better than another solution if the following conditions are satisfied:

$$X_1\ is\ better\ than\ X_2 : \begin{cases} f(X_1) > f(X_2) & if\ G(X_1) \le \varepsilon\ and\ G(X_2) \le \varepsilon \\ f(X_1) > f(X_2) & if\ G(X_1) = G(X_2) \\ G(X_1) < G(X_2) & otherwise \end{cases} \quad (13)$$

The ε parameter controls the size of feasible space. It is calculated by Eq. (14) (Takahama & Sakai, 2006).

$$\varepsilon(t) = \begin{cases} G_\theta\left(1 - \frac{t}{T_c}\right)^{cp} & if\ \ t \le T_c \\ 0 & otherwise \end{cases} \quad (14)$$

where t is the number of the current iteration. G_θ is the θ^{th} smallest constraint violation in the initial population. T_c is a parameter to truncate ε value and cp is a parameter to control the speed of reducing feasible space.

For constraint optimization problems, the comparisons of the SAR algorithm are done based on the ε-constrained method. Therefore, Eq. (6), 9, 10, and 11 are modified as follows:

$$X'_{i,j} = \begin{cases} \begin{cases} C_{k,j} + \mathrm{r1} \times (X_{i,j} - C_{k,j}) & if\ C_k\ is\ better\ than\ X_i \\ X_{i,j} + \mathrm{r1} \times (X_{i,j} - C_{k,j}) & otherwise \end{cases} & if\ r2 < SE\ \mathrm{or}\ j = j_{rand}\ j = 1, \cdots, D \\ X_{i,j} & otherwise \end{cases} \quad (15)$$

$$M_n = \begin{cases} X'_i & if\ X'_i\ is\ better\ than\ X_i \\ M_n & otherwise \end{cases} \quad (16)$$

$$X_i = \begin{cases} X'_i & if\ X'_i\ is\ better\ than\ X_i \\ X_i & otherwise \end{cases} \quad (17)$$

$$USN_i = \begin{cases} 0 & if\ X'_i\ is\ better\ than\ X_i \\ \mathrm{USN}_i + 1 & otherwise \end{cases} \quad (18)$$

where C_k or X'_i, are better than X_i if and only if Eq.13 is satisfied.

3.8. Restart strategy

Some constraint optimization problems have tremendously complex constraints. These constraints are multimodal and nonlinear and optimization algorithms can be simply converged in infeasible regions. To avoid this condition, a restart strategy has been introduced.

Firstly, a method is needed to recognize whether the population has been converged in a local optimum in infeasible regions. In this state, the entire population is infeasible. Also, similarities among them are excessive. For example, the standard deviation of degree of constraint violations or objective function values are very small. In the proposed restart strategy, if the standard deviation of degree of constraint violations is lower than a predefined value (α) and the population is infeasible, the proposed algorithm utilizes the restart strategy and the human and memory matrices are regenerated randomly.

Based on the above steps, the pseudo code of this algorithm has been presented in Algorithm 1 for solving a maximization constrained problem.

Algorithm 1 Pseudo code of SAR for constrained optimization.

1. Randomly initialize a population of 2 N solutions uniformly distributed in the range $[X_j^{min}, X_j^{max}]$, j = 1, . . . , D
2. Sort the solutions in the decreasing order and find the current best position (Xbest)
3. Use the first half of the sorted solutions for human positions (X) and the others for memory matrix (M)
4. Define the algorithm parameters and set USN_i = 0 where i = 1,. . .,N
5. **While** stop criterion is not satisfied **do**
6. **For** i = 1 to N **do**
7. Update the clues matrix (C) by Eq.4
8. **If** rand < 0.5 **do**
9. Apply the social phase and calculate the new position of the i^{th} human using Eq.15
10. **Else**
11. Apply the individual phase and calculate the new position of the i^{th} human using Eq.7
12. **End If**
13. Boundary control of the new position of the i^{th} human using Eq.8
14. Update the n^{th} memory and the position of the i^{th} human using Eq.16 and 17
15. Update USN_i using Eq.18
16. **If** $USN_i > MU$ and X_i is a feasible solution **do**
17. X_i is substituted with a random solution in the search space using Eq. (12) and $USN_i = 0$
18. **Else If** $USN_i > MU$ and X_i is an infeasible solution **do**
19. X_i is substituted with the solution with a minimum degree of constraint violation in the memory matrix and takes its place

$USN_i = 0$

20. **End If**
21. Apply the restart strategy
22. **End for**
23. Find the current best position and update Xbest
24. **End while**
25. Return Xbest

3.9. Control parameters of the SAR algorithm

SAR has two control parameters: *SE* (Social Effect) and *MU* (Maximum Unsuccessful Search Number). *SE* is used to control the effect of group members on each other in the social phase. *SE* is ranged in [0, 1]. Greater values of *SE* increase the convergence rate and also decrease the global search ability of the algorithm. Increasing the value of this parameter leads to more exploitation and less exploration. *MU* determines the maximum number of unsuccessful searches before leaving a clue. It is ranged in [0, $2 \times T_{max}$], where $2 \times T_{max}$ is the maximum number of search that is done by each human and T_{max} is the maximum number of iterations. For greater values of *MU*, humans will never leave the clues. On one hand, small values of this parameter lead to a group member finish his/her searching around the current clue and go to other locations before s/he can completely search around it. But on the other hand, large values of this parameter cause an increase in the searches performed around one clue and a decrease in the chances of searching in other regions. This parameter is directly related to the dimension of the problem. As the search space increases, *MU* is increased, too.

3.10. Conceptual Comparison of SAR with other metaheuristic algorithms

SAR is a population-based optimization algorithm. SAR consists of two phases including the social phase and the individual phase. The concepts of searching in these phases are different. Unlike the individual phase, new solutions are produced based on locations and objective functions of other solutions and each particle can move toward the other particles in the social phase. The comparison of SAR with PSO, DE, and TLBO are described as follows:

3.10.1. SAR versus PSO

Although a memory mechanism and a combination of social and individual information are utilized in these algorithms to produce new solutions, new solutions of PSO are produced by a combination of the global best position (social information) and the local best position (individual information). However, this information is used separately in SAR to produce two new solutions and the global best solution isn't considered by SAR. While SAR accepts new solutions which are better than current solutions, all new solutions are accepted by PSO. Furthermore, the memory update mechanisms of these methods are different. Also, PSO doesn't leave unimproved solutions to produce new solutions.

3.10.2. SAR versus DE

Both of these algorithms just accept new solutions which are better than current solutions. Although, the crossover mechanism of SAR in the social phase is similar to the binomial crossover of DE, the search strategy of DE is different from SAR. DE doesn't utilize the values of objective functions to produce new solutions and the previous solutions aren't considered by this algorithm. Furthermore, unlike SAR, DE doesn't leave unimproved solutions.

3.10.3. SAR versus TLBO

SAR and TLBO have two searching phases and in order to produce new solutions, objective function values are considered by them. Both of these algorithms just accept new solutions which are better than current solutions. Dissimilar to TLBO, SAR considers the previous solutions and leaves unimproved solutions after a certain number of unsuccessful objective function evaluations. Moreover, they produce new solutions using different methods.

In addition to the above differences, the boundary control strategy of SAR is different from those of PSO, DE, and TLBO.

3.11. Computational complexity

The computational complexity of SAR is presented here. The complexity of the population's initialization process is $O(2 \times N \times D)$, where *N* and *D* indicate the population size and the dimension of the problem. The proposed algorithm requires $O(2\ N \log(2\ N))$ times to sort the population in the initialization phase. In the worst case, the complexity of the social and individual phases are $O(N \times D)$ times. The abandoning clue process requires $O(N)$ times in the best and $O(N \times D)$ times in the worst case. Furthermore, the complexity of the restart process is $O(N \times D)$ in the worst case. Consequently, the computational complexity of SAR can be calculated as follows:

$$\begin{aligned} O(SAR) &= O(initialization) + T_{max} \times (O(Social\ phase) \\ &\quad + O(Individual\ phase) + O(Abandoning\ clue) \\ &\quad + O(Restart\ strategy)) \\ &= O(2 \times N \times D) + O(2Nlog(2N)) + T_{max} \times (O(N \times D) \\ &\quad + O(N \times D) + O(N \times D) + O(N \times D)) \end{aligned} \tag{19}$$

Table1
Characteristics of 13 benchmark constraint functions.

Problem name	Problem type	No. of variables	Function type	ρ (%)	LE	LI	NE	NI
g01	Min	13	Quadratic	0.0003	0	9	0	0
g02	Max	20	Nonlinear	99.9975	0	1	0	1
g03	Max	10	Nonlinear	0.0000	0	0	1	0
g04	Min	5	Quadratic	26.9356	0	0	0	6
g05	Min	4	Nonlinear	0.0000	0	2	3	0
g06	Min	2	Nonlinear	0.0064	0	0	0	2
g07	Min	10	Quadratic	0.0003	0	3	0	5
g08	Max	2	Nonlinear	0.864	0	0	0	2
g09	Min	7	Nonlinear	0.5256	0	0	0	4
g10	Min	8	Linear	0.0005	0	3	0	3
g11	Min	2	Quadratic	0.0000	0	0	1	0
g12	Min	3	Quadratic	0.0197	0	0	0	9^3
g13	Min	5	Nonlinear	0.0000	1	0	3	0

Overall, the total computational complexity of SAR is $O(T_{max} \times N \times D)$ in the worst case.

4. Experimental results and discussion

In this section, the performance of SAR is evaluated on two sets of benchmark test functions and 7 engineering design problems. The number of function evaluations is selected as a measure of computation time as a replacement for iterations. Furthermore, the feasible rates of the proposed algorithm for the following problems are reported. The feasible rate is the ratio of the number of feasible runs to total runs, where a feasible run is expressed as a run which at least one feasible solution is found by the optimization algorithm. In the following tables, "Best", "Mean", "Worst", and "Std" represent best, mean, worst, and standard deviation of function values, respectively. Also, "NA" means not available and the best values are highlighted in bold.

4.1. Benchmark test functions

The first set of benchmark test functions contains 13 benchmark constraint functions. The details of 13 benchmark constraint functions can be found in (Runarsson & Yao, 2000). The characteristics of these problems are presented in Table 1. ρ is the ratio of feasible search space to total search space. *LE*, *LI*, *NE*, and *NI* represent the number of linear equality, linear inequality, nonlinear equality, and nonlinear inequality constraints, respectively. The tolerance value (δ) for equality constraints was set to 10^{-6}.

The second set of benchmark test functions contains a set of 18 scalable constraint functions with 10 and 30 dimensions from IEEE CEC 2010 (Mallipeddi & Suganthan, 2010b). These functions are more complex than the 13 benchmark constraint functions. The characteristics of these problems are presented in Table 2. ρ is the ratio of feasible search space to total search space. *E* and *I* denote the number of equality and inequality constraints, respectively. The tolerance value (δ) for equality constraints was set to 10^{-4} according to the suggestion in (Mallipeddi & Suganthan, 2010b). For two sets of benchmark test functions and seven engineering design problems, 25 and 50 independent runs were performed by SAR, respectively.

4.2. Parameters setting

The parameters of the SAR algorithm were set as follows: $MU = 2 \times D$, for infeasible solutions, and $MU = 30 \times D$, for feasible solutions, where *D* is the number of variables. The population size of SAR (*N*) and the value of *SE* are presented in Table 3.

Table 3
The population size (N) and SE parameter for the proposed algorithm.

Test functions	N	*SE*
13 benchmark constraint functions	20	0.7
Engineering design problems	20	0.7
CEC 2010 functions with 10 dimension	25	0.4
CEC 2010 functions with 30 dimension	40	0.4

Table 2
Characteristics of CEC 2010 benchmark constraint functions.

Problem	Type of objective	Type of constraints	(No. of constraints)	Feasibility Region	
		E	I	10D	30D
C01	Non Separable	0	Non Separable (2)	0.997689	1
C02	Separable	Separable (1)	Separable (2)	0	0
C03	Non Separable	Non Separable (1)	0	0	0
C04	Separable	Separable (2), Non Separable (2)	0	0	0
C05	Separable	Separable (2)	0	0	0
C06	Separable	Rotated (2)	0	0	0
C07	Non Separable	0	Separable (1)	0.505123	0.503725
C08	Non Separable	0	Rotated (1)	0.379512	0.375278
C09	Non Separable	Separable (1)	0	0	0
C10	Non Separable	Rotated (1)	0	0	0
C11	Rotated	Non Separable (1)	0	0	0
C12	Separable	Non Separable (1)	Separable (1)	0	0
C13	Separable	0	Separable (2), Non Separable (1)	0	0
C14	Non Separable	0	Separable (3)	0.003112	0.006123
C15	Non Separable	0	Rotated (3)	0.00321	0.006023
C16	Non Separable	Separable (2)	Separable (2), Non Separable (1)	0	0
C17	Non Separable	Separable (1)	Non Separable (2)	0	0
C18	Non Separable	Separable (1)	Separable (1)	0.00001	0

Table 4
The statistical results of SAR for 13 benchmark constraint functions.

Problem name	Optimal	Best	Mean	Worst	Std	*MaxFEs*
g01	−15	−15	−15	−15	0.0E + 00	85,000
g02	−0.803619	−0.803619	−0.796242	−0.768629	9.78E-03	240,000
g03	−1	−1	−1	−1	3.16E-16	200,000
g04	−30665.53867	−30665.53867	−30665.53867	−30665.53867	0.0E + 00	30,000
g05	5126.4981	5126.4981	5126.4981	5126.4981	0.0E + 00	200,000
g06	−6961.813876	−6961.813876	−6961.813876	−6961.813876	0.0E + 00	30,000
g07	24.30621	24.30621	24.30621	24.30621	1.75E-12	200,000
g08	0.095825	−0.095825	−0.095825	−0.095825	7.08E-18	3500
g09	680.630057	680.630057	680.630057	680.630057	1.98E-13	40,000
g10	7049.248	7049.24802	7049.24802	7049.24803	3.12E-6	150,000
g11	0.75	0.75	0.75	0.75	0.0E + 00	40,000
g12	1	−1	−1	−1	0.0E + 00	6000
g13	0.0539498	0.0539498	0. 0,847,418	0. 4,388,507	1.07E-01	200,000

Table 5
The statistical results of the algorithms for 13 benchmark constraint functions.

Problem name	Feature	DEDS	HEAA	α SIMPLEX	BSAISA	DELC	GAFAT	BSA	SAR
g01	Best	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**
	Mean	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**	**−15.000**
	Std	1.3E-10	7.7E-11	6.4E-06	8.08E-16	6.5E-15	**0.0E + 00**	6.6E-16	**0.0E + 00**
	MaxFEs	225,000	200,000	303,162	84,630	150,000	150,000	99,300	85,000
g02	Best	**−0.803619**	−0.803582	**−0.803619**	−0.803599	**−0.803619**	−0.803173	−0.803255	**−0.803619**
	Mean	−0.78697	−0.758182	−0.784187	−0.787688	**−0.798877**	−0.77901	−0.79276	−0.796242
	Std	1.5E-02	3.2E-02	1.3E-02	1.14E-02	**7.7E-03**	2.8E-02	9.1E-03	9.79E-03
	MaxFEs	225,000	200,000	328,931	349,500	250,000	200,000	344,580	240,000
g03	Best	−1.0005	**−1.0000**	−1.0005	−1.000498	**−1.0000**	**−1.0000**	−1.000488	**−1.0000**
	Mean	−1.0005	**−1.0000**	−1.0005	−1.000481	**−1.0000**	**−1.0000**	−0.998905	**−1.0000**
	Std	1.9E-08	5.2E-15	8.5E-14	1.35E-05	2.1E-06	8.4E-07	2.66E-03	**3.16E-16**
	MaxFEs	225,000	200,000	310,968	58,560	200,000	200,000	348,960	200,000
g04	Best	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**
	Mean	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**	**−30665.539**
	Std	2.7E-11	7.4E-12	4.2E-11	1.09E-11	1E-11	**0.0E + 00**	1.11E-11	**0.0E + 00**
	MaxFEs	225,000	200,000	305,343	121,650	50,000	50,000	272,040	30,000
g05	Best	5126.497	**5126.498**	5126.497	5126.497	**5126.498**	**5126.498**	5126.497	**5126.498**
	Mean	5126.497	**5126.498**	5126.497	5126.497	**5126.498**	**5126.498**	5144.041	**5126.498**
	Std	0.0E + 00	9.3E-13	3.5E-11	5.85E-13	0.00051	**0.0E + 00**	39.9	**0.0E + 00**
	MaxFEs	225,000	200,000	308,389	238,410	200,000	50,000	299,220	200,000
g06	Best	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**
	Mean	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**	**−6961.814**
	Std	**0.0E + 00**	4.6E-12	1.3E-10	1.85E-12	7.3E-10	**0.0E + 00**	1.85E-12	**0.0E + 00**
	MaxFEs	225,000	200,000	293,367	89,550	20,000	20,000	111,450	30,000
g07	Best	**24.306**	**24.306**	**24.306**	24.307	**24.306**	**24.306**	24.308	**24.306**
	Mean	**24.306**	**24.306**	**24.306**	24.401	**24.306**	**24.306**	24.345	**24.306**
	Std	7.5E-07	1.9E-11	1.3E-04	1.02E-01	1.5E-06	**1.6E-14**	1.93E-02	1.75E-12
	MaxFEs	225,000	200,000	317,587	15,060	200,000	200,000	347,250	200,000
g08	Best	**−0.095825**	**−0.095825**	**−0.095825**	**−0.0958250**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**
	Mean	**−0.095825**	**−0.095825**	**−0.095825**	−0.086683	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**
	Std	4E-17	2.8E-17	3.8E-13	2.37E-02	1E-17	4E-17	2.82E-17	**7.08E-18**
	MaxFEs	225,000	200,000	306,248	30,930	5000	8000	73,440	3500
g09	Best	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**	**680.63**
	Mean	**680.63**	**680.63**	**680.63**	680.63	**680.63**	**680.63**	680.63	**680.63**
	Std	2.9E-13	5.8E-13	2.9E-10	9.11E-03	3.2E-12	5.6E-13	5.63E-04	**1.98E-13**
	MaxFEs	225,000	200,000	323,426	347,760	80,000	80,000	348,900	40,000
g10	Best	**7049.248**	**7049.248**	**7049.248**	7049.249	**7049.248**	**7049.248**	7049.278	**7049.248**
	Mean	7049.249	**7049.248**	**7049.248**	7081.242	**7049.248**	**7049.248**	7053.574	**7049.248**
	Std	1.4E-03	1.4E-05	4.7E-06	6.24E + 01	1.4E-04	2.2E-05	7.28	**3.12E-06**
	MaxFEs	225,000	200,000	316,037	346,980	200,000	180,000	340,020	150,000
g11	Best	0.7499	**0.75**	0.7499	0.7499	**0.75**	**0.75**	0.7499	**0.75**
	Mean	0.7499	**0.75**	0.7499	0.7499	**0.75**	**0.75**	0.7499	**0.75**
	Std	0.0E + 00	3.4E-16	4.9E-16	1.13E-16	**0.0E + 00**	5E-17	1.13E-16	**0.0E + 00**
	MaxFEs	225,000	200,000	308,125	87,870	50,000	20,000	113,250	40,000
g12	Best	**−1.000**	**−1.000**	**−1.000**	**−1.000**	**−1.000**	**−1.000**	**−1.000**	**−1.000**
	Mean	**−1.000**	**−1.000**	**−1.000**	**−1.000**	**−1.000**	**−1.000**	**−1.000**	**−1.000**
	Std	**0.0E + 00**	**0.0E + 00**	3.9E-10	**0.0E + 00**	**0.0E + 00**	9E-17	**0.0E + 00**	**0.0E + 00**
	MaxFEs	225,000	200,000	30,283	5430	5000	10,000	13,590	6000
g13	Best	0.053942	**0.0539498**	0.053942	0.0539415	**0.0539498**	**0.0539498**	0.053942	**0.0539498**
	Mean	0.053942	**0.0539498**	0.06677	0.103	**0.0539498**	**0.0539498**	0.1816986	0.0847418
	Std	1E-13	**1.5E-15**	6.9E-02	9.80E-02	4.7E-09	1.4E-09	1.5E-01	1.07E-01
	MaxFEs	225,000	200,000	311,144	349,800	200,000	90,000	347,400	200,000

Table 6
The statistical results of the algorithms for CEC 2010 benchmark constraint functions with 10D.

Problem	Feature	SAR	CMODE	BRGA	EABC	ICTLBO	RGA	ECHT-DE
C01	Mean	**−7.47E-01**	**−7.47E-01**	−7.29E-01	−7.16E-01	−7.44E-01	−7.21E-01	**−7.47E-01**
	Std	1.35E-03	2.35E-13	1.73E-02	2.69E-02	5.40E-03	2.62E-02	1.40E-03
C02	Mean	**−2.27E + 00**	−1.48E + 00*	3.35E + 00	−1.25E-01	−1.72E + 00	1.48E + 00	**−2.27E + 00**
	Std	1.53E-02	4.88E-01	1.26E + 00	1.58E + 00	8.10E-01	4.94E-01	6.70E-03
C03	Mean	**0.00E + 00**	2.84E + 00	2.19E + 13	2.45E + 12*	1.41E + 09	2.55E + 12*	**0.00E + 00**
	Std	0.00E + 00	4.23E + 00	3.86E + 13	1.01E + 12	6.00E + 09	2.36E + 07	0.00E + 00
C04	Mean	1.21E-01	−9.99E-06	3.99E + 00*	8.56E-01*	**−1.00E-05**	1.20E-03	**−1.00E-05**
	Std	2.88E-01	2.90E-08	7.37E + 00	3.01E + 00	1.70E-21	4.20E-03	0.00E + 00
C05	Mean	**−4.65E + 02**	−4.50E + 02*	1.17E + 02	3.65E + 02	−6.83E + 01	5.16E + 01*	−4.11E + 02
	Std	6.44E + 01	1.61E + 02	6.81E + 01	1.17E + 02	3.30E + 01	2.93E + 01	7.63E + 01
C06	Mean	−5.77E + 02	**−5.78E + 02**	1.87E + 02	4.38E + 02	−5.46E + 02	−1.18E + 02*	−5.62E + 02
	Std	1.67E-01	1.60E-02	5.70E + 01	8.59E + 01	3.20E + 01	−1.18E + 02	4.51E + 01
C07	Mean	3.19E-01	6.69E-15	1.26E + 01	7.16E + 01	**3.80E-24**	6.21E + 00	1.33E-01
	Std	1.10E + 00	8.95E-15	2.75E + 01	5.19E + 01	1.50E-23	9.30E + 00	7.28E-01
C08	Mean	**2.29E + 00**	8.94E + 00	2.72E + 02	4.11E + 02	1.72E + 01	6.21E + 00	6.16E + 00
	Std	4.35E + 00	3.98E + 00	4.56E + 02	9.36E + 02	2.60E + 01	9.30E + 00	6.45E + 00
C09	Mean	**0.00E + 00**	2.13E + 06*	7.89E + 02	2.02E + 12	3.56E + 05	2.72E + 08	1.47E-01
	Std	0.00E + 00	1.04E + 07	3.19E + 03	1.81E + 12	1.00E + 06	3.12E + 02	8.05E-01
C10	Mean	**0.00E + 00**	1.35E + 05*	5.60E + 02	1.75E + 12	1.32E + 06	1.04E + 09	1.71E + 00
	Std	0.00E + 00	1.61E + 06	7.13E + 02	2.58E + 12	6.50E + 06	5.84E + 04	7.66E + 00
C11	Mean	**−1.52E-03**	−7.70E-02*	−1.72E-01*	−1.23E + 00*	**−1.52E-03**	7.43E-01*	−4.40E-03*
	Std	5.04E-13	2.85E-02	1.90E + 00	3.04E + 00	4.80E-14	2.65E + 00	1.57E-02
C12	Mean	**−5.69E + 01**	−6.14E + 02*	1.67E + 01*	−1.80E + 02*	−5.01E + 01	−1.49E + 01*	−1.72E + 02*
	Std	1.34E + 02	2.74E + 02	3.71E + 01	2.76E + 02	1.40E + 02	1.67E + 02	2.21E + 02
C13	Mean	−6.81E + 01	−5.79E + 01	−6.15E + 01	−6.57E + 01	**−6.84E + 01**	−6.61E + 01	−6.51E + 01
	Std	9.46E-01	4.09E + 00	2.52E + 00	2.50E + 00	4.40E-14	2.23E + 00	2.37E + 00
C14	Mean	4.78E-01	**8.18E-09**	2.54E + 06	8.00E + 10	3.19E-01	4.93E + 04	7.02E + 05
	Std	1.32E + 00	1.64E-08	6.03E + 06	2.37E + 11	1.10E + 00	3.41E + 03	3.19E + 06
C15	Mean	2.17E + 13	1.20E + 02	8.20E + 06	2.57E + 13	**3.73E + 01**	8.92E + 08	2.34E + 13
	Std	4.22E + 13	3.48E + 02	1.57E + 07	2.86E + 13	9.40E + 01	4.12E + 04	5.30E + 13
C16	Mean	**0.00E + 00**	6.82E-05	9.89E-01	8.35E-02	4.27E-12	8.30E-01	3.93E-02
	Std	0.00E + 00	1.49E-04	8.58E-02	9.11E-02	2.10E-11	3.71E-01	4.28E-02
C17	Mean	1.44E + 01	**4.37E-02**	1.70E + 01	3.24E + 00	7.68E + 00	6.65E-01	1.12E-01
	Std	6.91E + 01	1.12E-01	9.55E + 00	6.83E + 00	3.70E + 01	1.03E + 00	3.32E-01
C18	Mean	**0.00E + 00**	5.75E + 00	2.59E + 00	3.47E + 02	2.59E + 00	2.39E + 00	**0.00E + 00**
	Std	0.00E + 00	2.64E + 02	3.92E + 00	3.71E + 02	1.20E + 01	1.93E + 00	0.00E + 00

Table 7
The Wilcoxon signed-rank test results for SAR and the compared algorithms on CEC 2010 functions with 10D.

SAR vs.	R^+	R^-	*p*-value	$\alpha = 0.05$	$\alpha = 0.1$
CMODE	118	35	4.17E-02	YES	YES
BRGA	157	14	1.72E-03	YES	YES
EABC	166	5	3.90E-04	YES	YES
ICTLBO	109	44	1.18E-01	NO	NO
RGA	147	24	6.63E-03	YES	YES
ECHT-DE	140	31	1.66E-02	YES	YES

The parameters of the ε-constrained method were set as follows: $T_c = 0.3 \times T_{max}$ for problems with equality constraints and zero for the other problems, where T_{max} indicates the maximum number of iterations. $\theta = 0.5$ and $cp = 0.3$. The predefined value of the restart strategy (α) was set to 10^{-7}.

4.3. Experiments on 13 benchmark constraint functions

The statistical results of SAR from 25 independent runs are shown in Table 4. The population size of SAR was equal to 20 for solving all the problems except g01 and g02. For these problems that have more variables, the population sizes of SAR were equal to 50 and 100, respectively. SAR successfully has found the global optimum in all test problems. So, the proposed algorithm can escape from the local optima in these problems.

These problems were also solved using several other methods such as DEDS (Zhang, Luo, & Wang, 2008), HEAA (Wang, Cai, Zhou, & Fan, 2009), the α SIMPLEX (Takahama & Sakai, 2005), BSAISA (Wang, Hu, Sun, Su, & Xia, 2018), DELC (Wang & Li, 2010), GAFAT (Zhao, Wang, Zeng, & Fan, 2012), and BSA (Wang, et al., 2018). The results of these methods are taken from their corresponding studies. The statistical results of these methods and SAR are presented in Table 5. The obtained results of SAR satisfied the constraints for all the problems. Hence, the feasible rates of the proposed algorithm are 100% for all the problems. According to Table 5, all the algorithms can find the global optimum for g01 and g04, but only the Std of SAR and GAFAT are zero. The *MaxFEs* of SAR is lower than GAFAT. Consequently, SAR outperforms the other algorithms for these problems. For g02, DEDS, α SIMPLEX, DELC, and SAR find the global optimum. DELC and SAR outperform the other algorithms in terms of mean and standard deviation. For g03, the performance of HEAA and SAR are better than the others. SAR performs better in terms of standard deviation. HEAA, DELC, GAFAT, and SAR find better mean values than the other algorithms for g05. The performance of the proposed algorithm is competitive for this problem. For g06, the standard deviations of DEDS, GAFAT, and SAR are zero and these algorithms are better than the others. SAR is the best algorithm because it requires the lowest *MaxFEs* in comparison to others. For g07, it can be seen that HEAA, GAFAT, and SAR obtain the best results. For g08, g09, g10, and g11, the best

Table 8
The statistical results of the algorithms for CEC 2010 benchmark constraint functions with 30D.

Problem	Feature	SAR	CMODE	BRGA	EABC	ICTLBO	RGA	ECHT-DE
C01	Mean	−8.07E-01	**−8.20E-01**	−6.94E-01	−7.31E-01	−8.18E-01	−7.72E-01	−8.00E-01
	Std	1.34E-02	3.30E-03	6.74E-02	4.88E-02	2.90E-03	3.20E-02	1.79E-02
C02	Mean	**−2.24E + 00**	9.75E-01	4.23E + 00	2.56E + 00	−3.83E-01	−3.96E-01	−1.99E + 00
	Std	3.96E-02	6.25E + 01	5.17E-01	9.43E-01	1.70E + 00	5.51E-01	2.10E-01
C03	Mean	**1.46E + 01**	2.18E + 01	2.45E + 13*	1.07E + 13*	2.13E + 11	3.57E + 12*	9.89E + 01
	Std	3.13E + 01	1.25E + 01	2.80E + 13	2.21E + 12	3.30E + 11	2.68E + 12	6.26E + 01
C04	Mean	5.64E + 00*	6.72E-04	4.34E + 00*	2.15E + 01*	1.59E-01	1.71E + 01*	**−1.03E-06**
	Std	9.30E + 00	4.24E-04	5.26E + 00	6.22E + 00	3.20E-01	1.36E + 01	9.01E-02
C05	Mean	**−2.90E + 02**	2.77E + 02*	2.64E + 02	3.72E + 02	−5.98E + 01	1.87E + 02	−1.06E + 02
	Std	6.20E + 01	2.03E + 02	7.07E + 01	7.89E + 01	8.80E + 00	7.90E + 01	1.67E + 02
C06	Mean	−4.59E + 02	−4.96E + 02*	2.68E + 02	4.74E + 02*	**−4.63E + 02**	−1.97E + 02*	−1.38E + 02
	Std	9.89E + 01	2.15E + 02	7.49E + 01	6.30E + 01	9.10E + 01	9.39E + 01	9.89E + 01
C07	Mean	3.19E-01	**5.24E-05**	7.45E + 02	1.33E + 02	2.88E + 01	3.65E + 01	1.33E-01
	Std	1.10E + 00	5.89E-05	1.57E + 03	2.06E + 02	4.60E + 01	2.27E + 01	7.29E-01
C08	Mean	4.61E + 01	**3.68E-01**	1.40E + 03	1.50E + 02	1.01E + 02	2.88E + 10	3.36E + 01
	Std	8.43E + 01	2.62E-01	1.87E + 03	7.15E + 01	1.20E + 02	4.23E + 07	1.11E + 02
C09	Mean	**2.46E + 00**	1.72E + 13*	2.15E + 05	1.61E + 13	1.01E + 07	2.72E + 08	4.24E + 01
	Std	8.80E + 00	1.07E + 13	4.68E + 05	9.29E + 12	3.50E + 07	3.12E + 04	1.38E + 02
C10	Mean	**1.63E + 01**	1.60E + 13*	1.60E + 04	1.50E + 13	6.01E + 09	6.99E + 08	5.34E + 01
	Std	2.40E + 01	7.00E + 12	1.89E + 04	9.77E + 12	2.30E + 10	6.13E + 03	8.83E + 01
C11	Mean	1.16E-03*	9.50E-03*	−9.57E-02*	−5.89E-01*	**−3.65E-04**	−1.68E-01*	2.60E-03*
	Std	5.27E-03	9.70E-03	2.44E-01	6.49E-01	4.90E-05	5.72E-01	6.00E-03
C12	Mean	1.01E + 00	−3.46E + 00*	−7.14E + 00*	5.07E + 01*	**−1.99E-01**	−1.25E + 02*	−2.51E + 01*
	Std	3.13E + 00	7.35E + 02	6.32E + 01	3.70E + 02	6.10E-05	1.54E + 02	1.36E + 02
C13	Mean	−6.65E + 01	−3.89E + 01	−5.63E + 01	−6.49E + 01	**−6.81E + 01**	−6.34E + 01	−6.46E + 01
	Std	1.18E + 00	2.17E + 00	2.32E + 00	1.38E + 00	7.70E-01	1.23E + 00	1.67E + 00
C14	Mean	**4.78E-01**	9.31E + 00	3.13E + 10	9.95E + 03	8.02E + 00	8.78E + 07	1.24E + 05
	Std	1.32E + 00	2.46E + 00	1.53E + 11	1.92E + 04	8.60E + 00	3.13E + 03	6.77E + 05
C15	Mean	7.73E + 13	1.51E + 13	5.78E + 06	3.79E + 13	**2.91E + 01**	7.99E + 09	1.94E + 11
	Std	3.64E + 13	8.26E + 12	1.11E + 07	3.44E + 13	3.60E + 01	5.13E + 04	4.35E + 11
C16	Mean	**0.00E + 00**	6.30E-02	1.05E + 00	8.21E-01	**0.00E + 00**	1.05E + 00	**0.00E + 00**
	Std	0.00E + 00	2.72E-02	2.51E-02	2.57E-01	0.00E + 00	3.99E-02	0.00E + 00
C17	Mean	1.99E + 02	3.12E + 02*	1.16E + 02	2.68E + 01	3.29E + 01	5.49E + 01	**2.75E-01**
	Std	1.77E + 02	2.75E + 02	4.77E + 01	1.63E + 01	1.30E + 02	1.98E + 01	3.78E-01
C18	Mean	3.02E-20	7.36E + 03	4.13E + 01	2.93E + 02	8.82E-04	4.40E + 01	**0.00E + 00**
	Std	8.88E-20	3.12E + 03	2.31E + 01	3.53E + 02	3.20E-03	1.61E + 01	0.00E + 00

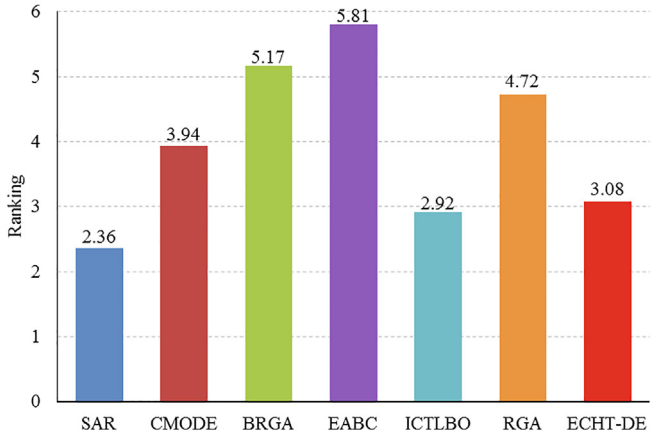

Fig. 2. Ranking of SAR and the compared algorithms by the Friedman's test on CEC 2010 functions with 10D.

results are achieved by SAR. The standard deviations of SAR are the lowest among the compared methods and it requires the lowest *MaxFEs* to obtain these results. Consequently, SAR is better than the other compared algorithms for these problems. For g12, it is easy for all the algorithms to find the best solution and most of them found it in all 50 runs. DELC, BSAISA, and SAR require the lowest number of function evaluations, respectively. For g13, although SAR found the global minimum, the results of HEAA, DELC, and GAFAT are significantly better than SAR. GAFAT requires the lowest *MaxFEs* to obtain these results and HEAA has the smallest standard deviations between all the algorithms.

4.4. Experiments on CEC 2010 benchmark functions

The performance of SAR on CEC 2010 benchmark functions is compared with six state-of-the-art constrained optimization algorithms: CMODE (Wang & Cai, 2012), BRGA (Abdul-Rahman, Munetomo, & Akama, 2013), EABC (Mezura-Montes & Velez-Koeppel, 2010), ICTLBO (Yu, Wang, & Wang, 2016), RGA

Table 9
The Wilcoxon signed-rank test results for SAR and the compared algorithms on CEC 2010 functions with 30D.

SAR vs.	R^+	R^-	*p*-value	$\alpha = 0.05$	$\alpha = 0.1$
CMODE	116	37	5.83E-02	NO	YES
BRGA	145.5	25.5	8.42E-03	YES	YES
EABC	145.5	25.5	8.42E-03	YES	YES
ICTLBO	80	73	8.50E-01	NO	NO
RGA	144.5	26.5	9.56E-03	YES	YES
ECHT-DE	106.5	64.5	3.49E-01	NO	NO

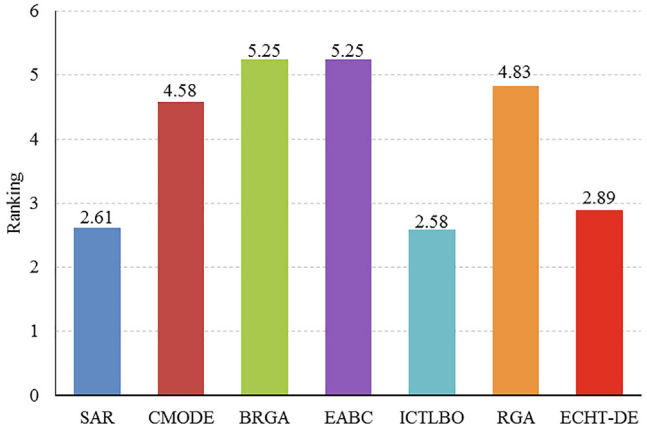

Fig. 3. Ranking of SAR and the compared algorithms by the Friedman's test on CEC 2010 functions with 30D.

(Saha, Datta, & Deb, 2010), and ECHT-DE (Mallipeddi & Suganthan, 2010a). The results of these optimization algorithms are taken from their corresponding studies. The results of SAR and the compared algorithms for 18 test functions with 10D and 30D are presented in Table 6 and Table 8, respectively. In these tables, the values with "*" indicate that the optimization algorithm could not find a feasible solution at the end of some runs. Table 6 illustrates SAR finds the best mean values on 11 functions. ICTLBO, and ECHT-DE obtain the best mean values on 5 benchmark functions. Among the compared algorithms, only the feasible rates of SAR and ICTLBO are 100% on all the problems.

To determine the differences between pairs of algorithms, the Wilcoxon signed-rank test is employed. The results of these statistical methods for CEC 2010 benchmark constraint functions with 10D are presented in Table 7. According to this table, SAR is remarkably better than CMODE, BRGA, EABC, RGA, and ECHT-DE at a 5% significance level for CEC 2010 functions with 10 dimensions. There is no significant difference between SAR and ICTLBO. The mean values of SAR are better, equal and worse than ICTLBO on 11, 1 and 6 functions, respectively.

Furthermore, the rankings of multiple algorithms are obtained by the Friedman test and present in Fig. 2. This figure shows that SAR gets the first ranking between the compared algorithms, followed by ECHT-DE, ICTLBO, and CMODE, respectively.

According to Table 8, the performance of SAR is very competitive with the other compared algorithms on CEC 2010 functions with 30D. SAR, CMODE, ICTLBO, and ECTE-DE obtain the best mean values on 7, 3, 6 and 4 test functions. However, BRGA, EABC, and RGA don't achieve any best mean values. Among the compared algorithms, only the feasible rates of ICTLBO are 100% on all the problems. While SAR obtains a 100% feasible rates on 16 functions, the feasible rates of SAR are 68% and 88% on C04 and C11, respectively.

The results of the Wilcoxon signed-rank test for CEC 2010 benchmark constraint functions with 30D are presented in Table 9. As shown in this table, SAR significantly outperforms BRGA, EABC, and RGA at a 5% significance level and outperforms CMODE at a 10% significance level. There are no significant differences between SAR and ICTLBO and ECHT-DE.

According to the results of the Friedman test are shown in Fig. 3, ICTLBO is ranked the best and SAR achieves a slightly higher rank than this algorithm on CEC 2010 functions with 30D.

4.5. Application to engineering design problems

The previous comparative results reveal that SAR is a competitive and efficient algorithm for constrained optimization problems. In this subsection, to validate the ability of SAR to solve the optimization problems in real-world applications, the proposed algorithm is applied to solve 7 real-world engineering design problems including speed reducer design (Golinski, 1973), three-bar truss design (Nowacki, 1973), pressure vessel design (Sandgren, 1990), welded beam design (Ragsdell & Phillips, 1976), spring design (Belegundu, 1983), tubular column design (Rao, 1996) and reinforced concrete beam design (Amir & Hasegawa, 1989). The details of these problems can be found in their corresponding literature. The results of SAR are obtained by performing 50 independent runs and compared with some state-of-the-art optimization algorithms. The results of these methods are taken from their corresponding studies. Table 10 presents the best values of design variables, objective function, and constraints obtained by SAR for the engineering design problems.

Fig. 4 shows the convergence curves of $f(x)$-$f(x^*)$ over *MaxFEs* for the average run of SAR regarding the constrained engineering problems, where x^* is the best found solution.

Table 10
Optimal design variables, objective function and constraints obtained by SAR for the engineering design problems.

	Speed Reducer	Three-bar Truss	Pressure vessel	Welded beam	Spring design	Tubular column	Reinforced concrete beam
x1	3.5	0.7886751	0.8125	0.2057296	0.0516893	5.451156234	6.32
x2	0.7	0.4082483	0.4375	3.4704887	0.3567232	0.291965477	34
x3	17		42.098446	9.0366239	11.288648		8.5
x4	7.3		176.6365958	0.2057296			
x5	7.7153199						
x6	3.3502147						
x7	5.2866545						
f(x)	2994.471066	263.895843	6059.714335	1.7248523	0.0126652	26.531328	359.208
g1	−0.074	0	−4.40E-16	−1.50E-05	−2.10E-12	−2.77E-10	0
g2	−0.198	−1.464	−0.036	−2.90E-05	−1.40E-11	−7.25E-12	−0.2241
g3	−0.499	−0.536	−1.70E-09	0	−4.054		
g4	−0.905		−63.363	−3.433	−0.728		
g5	−2.00E-12			−0.081			
g6	−4.30E-12			−2.30E-01			
g7	−0.702			−1.90E-05			
g8	−2.20E-13						
g9	−0.583						
g10	−0.051						
g11	−3.10E-12						

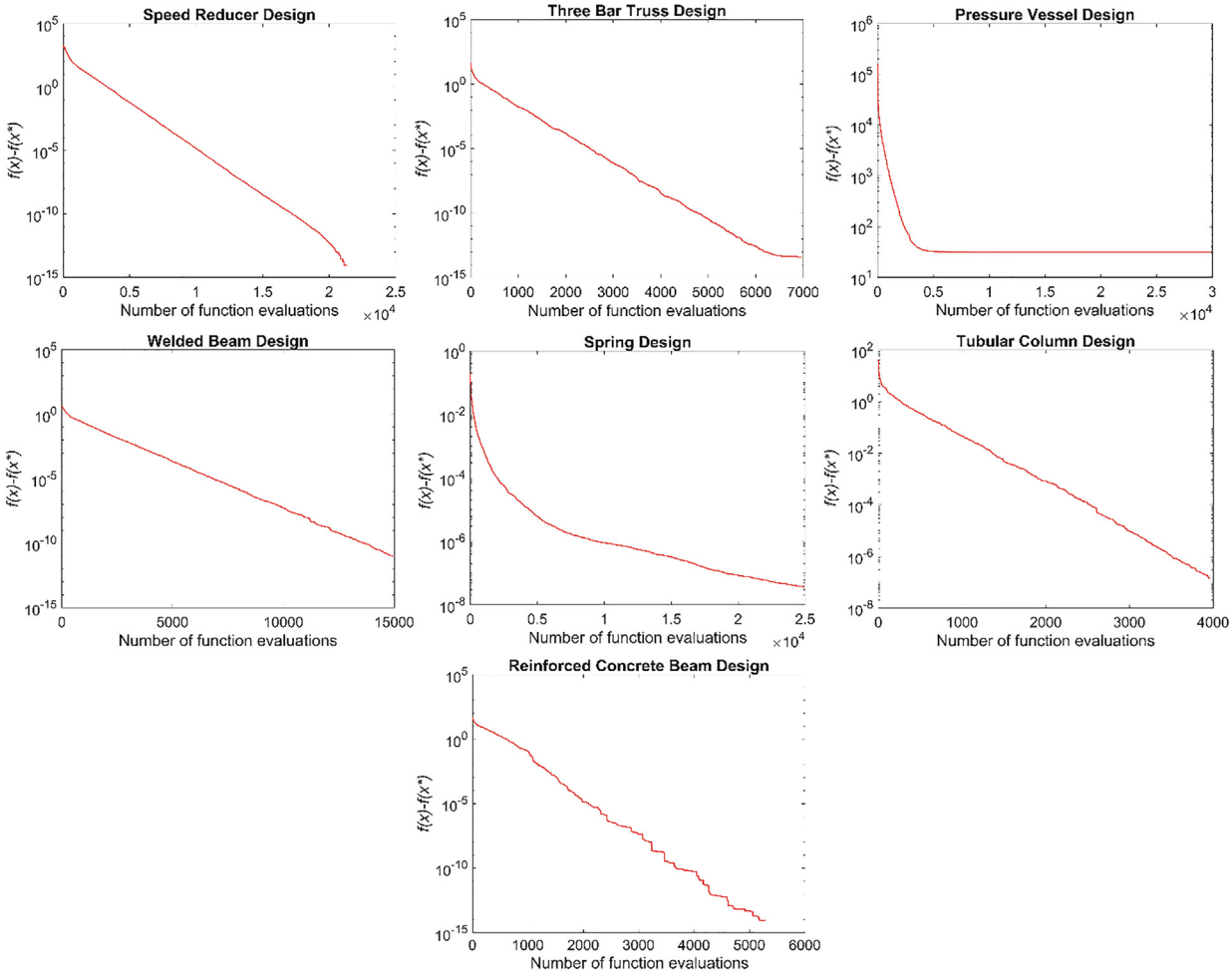

Fig. 4. The convergence curves of SAR for the constrained engineering problems.

Table 11
Comparison of the statistical results for the speed reducer design problem.

Method	Best	Mean	Worst	Std	*MaxFEs*
SCA	2994.744	3001.758	NA	4.00E + 00	54,456
(μ + λ)-ES	2996.348	2996.348	NA	0.00E + 00	30,000
MDE	2996.356689	2996.36722	2996.390137	8.20E-03	24,000
SiC-PSO	2996.348165	2996.3482	NA	0.00E + 00	24,000
DEDS	**2994.471066**	**2994.471066**	**2994.471066**	3.58E-12	30,000
HEAA	2994.499107	2994.613368	2994.752311	7.00E-02	40,000
PSO-DE	2996.348167	2996.348174	2996.348204	6.40E-06	54,350
ABC	2997.058412	2997.058412	NA	0.00E + 00	30,000
εDE-PCGA	**2994.471066**	**2994.471066**	**2994.471066**	1.16E-10	20,000
IAFOA	2996.347898	2996.348356	2996.348069	3.52E-05	240,000
SAR	**2994.471066**	**2994.471066**	**2994.471066**	**0.00E + 00**	22,000

Table 12
Comparison of the statistical results for the three-bar truss design problem.

Method	Best	Mean	Worst	Std	*MaxFEs*
DEDS	**263.895843**	**263.895843**	263.895849	9.70E-07	15,000
HEAA	**263.895843**	263.895865	263.896099	4.9E-05	15,000
PSO-DE	**263.895843**	**263.895843**	**263.895843**	4.50E-10	17,600
BAT	263.896248	263.90614	263.9024677	3.53E-03	15,000
CS	263.97156	264.0669	NA	9.00E-05	15,000
BSA	**263.895843**	**263.895843**	263.895845	2.64E-07	13,720
εDE-PCGA	**263.895843**	**263.895843**	**263.895843**	**2.13E-14**	15,000
CSA	**263.895843**	**263.895843**	**263.895843**	1.01E-10	25,000
BSAISA	**263.895843**	**263.895843**	**263.895843**	5.75E-13	8400
SAR	**263.895843**	**263.895843**	**263.895843**	2.68E-14	7000

Table 13
Comparison of the statistical results for the pressure vessel design problem.

Method	Best	Mean	Worst	Std	*MaxFEs*
GA-DT	6059.9463	6177.2533	6469.322	1.31E + 02	80,000
CPSO	6061.0777	6147.1332	6363.8041	8.65E + 01	30,000
HPSO	**6059.7143**	6099.9323	6288.677	8.62E + 01	81,000
CMA-ES	**6059.714335**	6170.250553	6410.08676	1.40E + 02	30,018
DELC	**6059.7143**	**6059.7143**	**6059.7143**	**2.10E-11**	30,000
BSA-SAε	**6059.7143**	6074.3682	6116.7804	1.71E + 01	80,000
ABC	6059.714736	6245.308144	NA	2.05E + 02	30,000
GDA	6059.8391	6149.7276	6823.6024	2.11E + 02	20,000
BAT	**6059.7143**	6179.13	6318.95	1.37E + 02	20,000
BSA	6059.715	6221.2861	6771.5969	2.03E + 02	60,000
CSA	6059.7144	6342.4991	7332.8416	3.85E + 02	250,000
BSAISA	**6059.7143**	6418.1935	7198.0054	3.04E + 02	16,320
SAR	**6059.714335**	6091.32594	6410.0868	8.03E + 01	30,000

Table 14
Comparison of the statistical results for the welded beam design problem.

Method	Best	Mean	Worst	Std	*MaxFEs*
SAPA	1.748309	1.771973	1.785835	1.12E-02	900,000
GA-DT	1.728226	1.792654	1.993408	7.47E-02	80,000
(μ + λ)-ES	**1.724852**	1.777692	NA	8.80E-02	30,000
CPSO	1.728024	1.748831	1.782143	1.29E-02	30,000
BSA-SAε	**1.724852**	**1.724852**	**1.724852**	8.11E-10	80,000
IACO	1.724918	1.729752	1.775961	9.20E-03	17,600
ABC	**1.724852**	1.741913	NA	3.10E-02	30,000
BSA	**1.724852**	**1.724852**	1.724854	2.35E-07	45,480
TEO	1.725284	1.76804	1.931161	5.82E-02	20,000
BSAISA	**1.724852**	1.724852	1.724854	2.96E-07	29,000
IGWO	1.725	1.7255	1.7263	4.72E-04	50,000
IAFOA	1.724856	1.724856	1.724856	8.99E-07	240,000
DDB-BC	**1.724852**	1.724855	1.72489	6.97E-06	18,000
SAR	**1.7248523**	**1.7248523**	**1.7248523**	**2.22E-11**	15,000

Table 15
Comparison of the statistical results for the spring design problem.

Method	Best	Mean	Worst	Std	*MaxFEs*
MDE	**0.012665**	**0.012666**	0.012674	2.00E-06	24,000
CPSO	0.012675	0.01273	0.012924	5.20E-05	23,000
CEDE	0.0126702	0.0126703	0.012679	2.70E-05	204,800
AATM	0.0126683	0.0127081	0.0128614	4.50E-05	25,000
BSA-SAε	**0.0126652**	**0.0126653**	**0.0126658**	1.62E-07	80,000
GSA	0.0128739	0.0134389	0.0142117	1.34E-02	30,000
ABC	**0.012665**	0.012709	NA	1.28E-02	30,000
SSOC	**0.012665**	0.012765	0.012868	9.29E-05	25,000
CSA	**0.0126652**	**0.012666**	0.0126702	1.36E-06	50,000
TEO	**0.012665**	0.012685	0.012715	4.41E-06	300,000
MGWO	0.0126696	0.0126799	0.0127057	1.10E-05	30,000
IGWO	0.012667	0.012691	0.012718	1.97E-05	50,000
IAFOA	**0.012665**	0.012673	0.012688	3.02E-06	240,000
SAR	**0.0126652**	**0.0126653**	0.0126675	**1.26E-07**	25,000

4.5.1. Speed reducer design

Several methods have been utilized to solve the speed reducer design problem including SCA (Ray & Liew, 2003), (μ + λ)-ES (Mezura-Montes & Coello, 2005), MDE (Mezura-Montes, Coello, & Velázquez-Reyes, 2006), SiC-PSO (Cagnina, Esquivel, & Coello, 2008), DEDS (Zhang, et al., 2008), HEAA (Wang, Cai, Zhou et al., 2009), PSO-DE (Liu, Cai, & Wang, 2010), ABC (Akay & Karaboga, 2012), εDE-PCGA (Yi, Li, Gao, Zhou, & Huang, 2016) and IAFOA (Wu, Liu, Tian, Zhang, & Xiao, 2018). The statistical results of these methods and SAR are shown in Table 11.

With regard to this table, only DEDS, εDE-PCGA, and SAR can find the best solution for this problem. DEDS, εDE-PCGA, and SAR obtain better mean and worst values than the other methods. The number of function evaluations and the standard deviation of SAR are lower than DEDS, thus SAR outperforms DEDS for this problem. Although *MaxFEs* of εDE-PCGA is slightly lower than SAR, the standard deviation of SAR is significantly better than εDE-PCGA. In comparison with the other algorithms, SAR achieves the best mean, worst and standard deviation and the lowest *MaxFEs*.

4.5.2. Three-bar truss design

Table 12 illustrates the statistical results for SAR and DEDS (Zhang, et al., 2008), HEAA (Wang, Cai, Zhou et al., 2009), PSO-DE (Liu, et al., 2010), BAT (Yang & Hossein Gandomi, 2012), CS (Gandomi, et al., 2013), BSA (Wang, et al., 2018), εDE-PCGA (Yi, et al., 2016), CSA (Askarzadeh, 2016), and BSAISA (Wang, et al., 2018) algorithms for solving the three-bar truss design problem. Most of the algorithms can find the best solution for this problem. εDE-PCGA, BSAISA, and SAR have demonstrated better

Table 16
Comparison of the statistical results for the tubular column design problem.

Method	Best	Mean	Worst	Std	*MaxFEs*
EM	26.5338	NA	NA	7.70E-02	17,308
HEM	26.53227	NA	NA	3.50E-03	25,136
AFSF	26.53351	NA	NA	5.20E-03	9223
CS	26.53217	26.53504	26.53972	1.93E-03	15,000
KH	26.5314	26.543	26.6475	1.80E-02	10,000
SAR	**26.531328**	**26.531328**	**26.531328**	**1.51E-07**	4000

Table 17
Comparison of the statistical results for the reinforced concrete beam design problem.

Method	Best	Mean	Worst	Std	*MaxFEs*
GHN-ALM	362.2455	NA	NA	NA	NA
GHN-EP	362.0065	NA	NA	NA	NA
GA	366.1459	371.5417	NA	NA	100,000
FLC-AHGA	364.8541	365.8046	NA	NA	100,000
FA	**359.208**	460.706	669.15	8.07E + 01	25,000
BB-BC	**359.208**	360.5401	362.634	1.65E + 00	50,000
DBB-BC	**359.208**	**359.208**	**359.208**	5.78E-14	17,000
SAR	**359.208**	**359.208**	**359.208**	**0.00E + 00**	6000

performance in terms of the standard deviation compared to the other techniques, but SAR requires the lowest *MaxFEs*.

4.5.3. Pressure vessel design

For this problem, SAR is compared with GA-DT (Coello & Montes, 2002), CPSO (He & Wang, 2007a), HPSO (He & Wang, 2007b), CMA-ES (De Melo & Iacca, 2014), DELC (Wang & Li, 2010), BSA-SAε (Zhang, Lin, Gao, & Li, 2015), ABC (Akay & Karaboga, 2012), GDA (Baykasoglu, 2012), BAT (Yang & Hossein Gandomi, 2012), BSA (Wang, et al., 2018), CSA (Askarzadeh, 2016), and BSAISA (Wang, et al., 2018) algorithms. The statistical results of these algorithms are given in Table 13.

Most of the algorithms can find the best function value. DELC is superior to the other algorithms in terms of mean, worst, standard deviation, and the number of function evaluations. Between the remaining algorithms, BSA-SAε, SAR, and CPSO have better performances for this problem, respectively.

4.5.4. Welded beam design

Several methods have been applied to solve the welded beam design problem such as SAPA (Coello, 2000), GA-DT (Coello & Montes, 2002), (μ + λ)-ES (Mezura-Montes & Coello, 2005), CPSO (He & Wang, 2007a), BSA-SAε (Zhang, et al., 2015), IACO (Kaveh & Talatahari, 2010), ABC (Akay & Karaboga, 2012), BSA (Wang, et al., 2018), TEO (Kaveh & Dadras, 2017), BSAISA (Wang, et al., 2018), IGWO (Long, Jiao, Liang, & Tang, 2018), IAFOA (Wu, et al., 2018) and DDB-BC (Prayogo, et al., 2018). The statistical results for SAR and these methods are shown in Table 14. SAR and most of the algorithms can find the optimal design of this problem. SAR requires the lowest *MaxFEs* between all the algorithms to find the optimal solution. Furthermore, the mean value, worst value, and standard deviation of this algorithm are better than the others.

4.5.5. Spring design

SAR is compared with MDE (Mezura-Montes, et al., 2006), CPSO (He & Wang, 2007a), CEDE (Huang, Wang, & He, 2007), AATM (Wang, Cai, & Zhou, 2009), BSA-SAε (Zhang, et al., 2015), GSA (Rashedi, Nezamabadi-Pour, & Saryazdi, 2009), ABC (Akay & Karaboga, 2012), SSOC (Cuevas & Cienfuegos, 2014), CSA (Askarzadeh, 2016), TEO (Kaveh & Dadras, 2017), MGWO (Kumar & Kumar, 2017), IGWO (Long, et al., 2018) and IAFOA (Wu, et al., 2018) algorithms for the spring design problem. The statistical results of these algorithms are summarized in Table 15.

The mean, worst, and standard deviation of SAR are lower than the other methods except BSA-SAε. SAR and BSA-SAε have similar performance for this problem. Furthermore, SAR requires lower *MaxFEs* to obtain these results and it has the fastest convergence rate between all the methods.

4.5.6. Tubular column design

The results of EM (Rocha & Fernandes, 2009), HEM (Rocha & Fernandes, 2009), AFSF (Rocha, Costa, & Fernandes, 2012), CS (Gandomi, et al., 2013) and KH (Gandomi & Alavi, 2012) algorithms are compared with SAR for the Tubular column design problem. Their correspondingly statistical results are summarized in Table 16. SAR requires the lowest *MaxFEs* among all the algorithms to find the best solution for this problem. Also, it can be seen that the standard deviation of the results obtained by SAR is better than the other techniques.

4.5.7. Reinforced concrete beam design

Several methods have been utilized to solve the reinforced concrete beam design problem including GHN-ALM and GHN-EP (Shih & Yang, 2002), GA and FLC-AHGA (Yun, 2005), FA (Gandomi, et al., 2011), BB-BC and DDB-BC (Prayogo, et al., 2018). The statistical results of these methods and SAR are listed in Table 17. FA, BB-BC, DBB-BC, and SAR are able to find the best function value. DBB-BC and SAR have the lowest standard deviations for this problem, however, SAR requires lower *MaxFEs* than DBB-BC.

According to the aforementioned comparison results, it can be seen that SAR has demonstrated better or competitive performance compared to the studied algorithms for solving the engineering design problems.

5. Conclusion

In the present study, a new metaheuristic optimization method namely the Search and Rescue optimization algorithm (SAR) is proposed to solve constrained engineering optimization problems. SAR imitates the explorations behavior of humans during search and rescue operations. The proposed algorithm consists of two phases including the social phase and the individual phase and the concepts of searching in these phases are different. In SAR, constraints are handled using the ε-constrained method. Moreover, to deal with complicated constraints, a straightforward restart strategy is introduced. The performance of SAR is evaluated on two sets

of benchmark constraint functions and 7 engineering design problems. The results obtained by SAR are compared with some other optimization algorithms. As mentioned, based on the No Free Lunch Theorem (NFL), there is no optimization algorithm which works well on all classes of optimization problems. However, the results of the statistical comparisons indicate that SAR obtains better or highly competitive performance on most of the studied constraint problems.

The implementation of SAR is rather easy and it can be simply applied in various kinds of optimization problems. In future works, SAR will be applied to solve large scale and more complex constrained optimization problems.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abdul-Rahman, O. A., Munetomo, M., & Akama, K. (2013). An adaptive parameter binary-real coded genetic algorithm for constraint optimization problems: Performance analysis and estimation of optimal control parameters. *Information sciences, 233*, 54–86.

Akay, B., & Karaboga, D. (2012). Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of intelligent manufacturing, 23*, 1001–1014.

Amir, H. M., & Hasegawa, T. (1989). Nonlinear mixed-discrete structural optimization. *Journal of Structural engineering, 115*, 626–646.

Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. *Computers & structures, 169*, 1–12.

ASTM (2005). *F1993: Standard Classification of Human Search and Rescue Resources. In. West*. Conshohocken, PA, USA: ASTM International.

Baykasoglu, A. (2012). Design optimization with chaos embedded great deluge algorithm. *Applied Soft Computing, 12*, 1055–1067.

Belegundu, A. D. (1983). *A study of mathematical programming methods for structural optimization*. USA: University of Iowa.

Cagnina, L. C., Esquivel, S. C., & Coello, C. A. C. (2008). *Solving engineering optimization problems with the simple constrained particle swarm optimizer., Informatica*, 32.

Cheng, M. Y., & Prayogo, D. (2017). A novel fuzzy adaptive teaching–learning-based optimization (FATLBO) for solving structural optimization problems. *Engineering with Computers, 33*, 55–69.

Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry, 41*, 113–127.

Coello, C. A. C., & Montes, E. M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics, 16*, 193–203.

Couzin, I. D., Krause, J., Franks, N. R., & Levin, S. A. (2005). Effective leadership and decision-making in animal groups on the move. *Nature, 433*, 513–516.

Cuevas, E., & Cienfuegos, M. (2014). A new algorithm inspired in the behavior of the social-spider for constrained optimization. *Expert Systems with Applications, 41*, 412–425.

De Melo, V. V., & Iacca, G. (2014). A modified covariance matrix adaptation evolution strategy with adaptive penalty function and restart for constrained optimization. *Expert Systems with Applications, 41*, 7077–7094.

Deb, K. (2012). *Optimization for engineering design: Algorithms and examples: PHI*. Learning Pvt. Ltd..

Du, K. L., & Swamy, M. (2016). *Particle swarm optimization. In Search and optimization by metaheuristics*. Springer.

Gandomi, A. H., & Alavi, A. H. (2012). Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation, 17*, 4831–4845.

Gandomi, A. H., Yang, X.-S., & Alavi, A. H. (2011). Mixed variable structural optimization using firefly algorithm. *Computers & structures, 89*, 2325–2336.

Gandomi, A. H., Yang, X.-S., & Alavi, A. H. (2013). Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with computers, 29*, 17–35.

Gharebaghi, S. A., Kaveh, A., & Asl, M. A. (2017). A new meta-heuristic optimization algorithm using star graph. *SMART STRUCTURES AND SYSTEMS, 20*, 99–114.

Golinski, J. (1973). An adaptive optimization system applied to machine synthesis. *Mechanism and Machine Theory, 8*, 419–436.

He, Q., & Wang, L. (2007a). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence, 20*, 89–99.

He, Q., & Wang, L. (2007b). A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Applied mathematics and computation, 186*, 1407–1422.

Huang, F.-Z., Wang, L., & He, Q. (2007). An effective co-evolutionary differential evolution for constrained optimization. *Applied mathematics and computation, 186*, 340–356.

Kaveh, A., & Dadras, A. (2017). A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Advances in Engineering Software, 110*, 69–84.

Kaveh, A., & Talatahari, S. (2010). An improved ant colony optimization for constrained engineering design problems. *Engineering Computations, 27*, 155–182.

Kennedy, J., & Eberhart, R. (1995). *Particle swarm optimization, Vol. 4*, 1942–1948.

Kumar, V., & Kumar, D. (2017). An astrophysics-inspired Grey wolf algorithm for numerical optimization and its application to engineering design problems. *Advances in Engineering Software, 112*, 231–254.

Liu, H., Cai, Z., & Wang, Y. (2010). Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing, 10*, 629–640.

Long, W., Jiao, J., Liang, X., & Tang, M. (2018). Inspired grey wolf optimizer for solving large-scale function optimization problems. *Applied Mathematical Modelling, 60*, 112–126.

Mallipeddi, R., & Suganthan, P. N. (2010a). Differential evolution with ensemble of constraint handling techniques for solving CEC 2010 benchmark problems. In IEEE congress on evolutionary computation (pp. 1-8): IEEE.

Mallipeddi, R., & Suganthan, P. N. (2010). *Problem definitions and evaluation criteria for the CEC 2010 competition on constrained real-parameter optimization* (p. 24). Singapore: Nanyang Technological University.

Kentucky Emergency Management. (2020). SAR Field Search Methods: Search Techniques Used by Trained Teams in the Field. In: Kentucky Emergency Management. Available from: URL: https://kyem.ky.gov/Who%20We%20Are/Documents/SAR%20Field%20Search%20Methods.pdf. Accessed June 03 2020.

Mezura-Montes, E., & Coello, C. A. C. (2005). In *Useful infeasible solutions in engineering optimization with evolutionary algorithms* (pp. 652–662). Springer.

Mezura-Montes, E., Coello, C. C., & Velázquez-Reyes, J. (2006). Increasing successful offspring and diversity in differential evolution for engineering design. In Proceedings of the seventh international conference on adaptive computing in design and manufacture (ACDM 2006) (pp. 131-139).

Mezura-Montes, E., & Velez-Koeppel, R. E. (2010). Elitist artificial bee colony for constrained real-parameter optimization. In IEEE congress on evolutionary computation (pp. 1-8): IEEE.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.

Nowacki, H. (1973). Optimization in pre-contract ship design. *Computer Applications in the Automation of Shipyard Operation and Ship Design, 2*, 327–338.

Osman, I. H., & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research, 63*, 513–623.

OSARVA Senior Search and Rescue Trainers (2016). Search Techniques Manual. In: Ontario Search and Rescue Volunteer Association.

Prayogo, D., Cheng, M.-Y., Wu, Y.-W., Herdany, A. A., & Prayogo, H. (2018). Differential Big Bang-Big Crunch algorithm for construction-engineering design optimization. *Automation in Construction, 85*, 290–304.

Ragsdell, K., & Phillips, D. (1976). Optimal design of a class of welded structures using geometric programming. *Journal of Engineering for Industry, 98*, 1021–1025.

Rao, S. S. (1996). Engineering Optimization, Theory and Practice 3rd. Edition, chapter 1 Introduction to Optimization. Wiley and Sons.

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information sciences, 179*, 2232–2248.

Ray, T., & Liew, K. M. (2003). Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE transactions on evolutionary computation, 7*, 386–396.

Rocha, A. M. A., Costa, M. F. P., & Fernandes, E. M. (2012). In *An artificial fish swarm filter-based method for constrained global optimization* (pp. 57–71). Springer.

Rocha, A. M. A., & Fernandes, E. M. (2009). Hybridizing the electromagnetism-like algorithm with descent search for solving engineering design problems. *International journal of computer mathematics, 86*, 1932–1946.

Rubio-Marín, R. (2014). *Human rights and immigration*. Oxford University Press.

Runarsson, T. P., & Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE transactions on evolutionary computation, 4*, 284–294.

Saha, A., Datta, R., & Deb, K. (2010). Hybrid gradient projection based genetic algorithms for constrained optimization. In IEEE congress on evolutionary computation (pp. 1-8): IEEE.

Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design, 112*, 223–229.

Shabani, A., Asgarian, B., & Salido, M. (2019a). Search and Rescue Optimization Algorithm for Size Optimization of Truss Structures with Discrete Variables. *International Journal of Numerical Methods in Civil Engineering, 3*, 28–39.

Shabani, A., Asgarian, B., Gharebaghi, S. A., Salido, M. A., & Giret, A. (2019b). A New Optimization Algorithm Based on Search and Rescue Operations. *Mathematical Problems in Engineering, 2019*.

Shih, C., & Yang, Y. (2002). Generalized Hopfield network based structural optimization using sequential unconstrained minimization technique with additional penalty strategy. *Advances in Engineering Software, 33*, 721–729.

Takahama, T., & Sakai, S. (2005). Constrained optimization by applying the/spl alpha/constrained method to the nonlinear simplex method with mutations. *IEEE transactions on evolutionary computation, 9*, 437–451.

Takahama, T., & Sakai, S. (2006). Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. In 2006 IEEE International Conference on Evolutionary Computation (pp. 1-8): IEEE.

Torres-Jiménez, J., & Pavón, J. (2014). Applications of metaheuristics in real-life problems. Springer.
Wang, H., Hu, Z., Sun, Y., Su, Q., & Xia, X. (2018). Modified Backtracking Search Optimization Algorithm Inspired by Simulated Annealing for Constrained Engineering Optimization Problems. *Computational Intelligence and Neuroscience, 2018*, 1–27. https://doi.org/10.1155/2018/9167414.
Wang, L., & Li, L.-P. (2010). An effective differential evolution with level comparison for constrained engineering design. *Structural and Multidisciplinary Optimization, 41*, 947–963.
Wang, Y., & Cai, Z. (2012). Combining multiobjective optimization with differential evolution to solve constrained optimization problems. *IEEE transactions on evolutionary computation, 16*, 117–134.
Wang, Y., Cai, Z., & Zhou, Y. (2009). Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization. *International Journal for Numerical Methods in Engineering, 77*, 1501–1534.
Wang, Y., Cai, Z., Zhou, Y., & Fan, Z. (2009). Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. *Structural and Multidisciplinary Optimization, 37*, 395–413.
Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*, 67–82.
Wu, J., Wang, Y.-G., Burrage, K., Tian, Y.-C., Lawson, B., & Ding, Z. (2020). An improved firefly algorithm for global continuous optimization problems. *Expert Systems with Applications* 113340.
Wu, L., Liu, Q., Tian, X., Zhang, J., & Xiao, W. (2018). A new improved fruit fly optimization algorithm IAFOA and its application to solve engineering optimization problems. *Knowledge-Based Systems, 144*, 153–173.
Yadav, A., & Kumar, N. (2020). Artificial electric field algorithm for engineering optimization problems. *Expert Systems with Applications, 149*, 113308.
Yang, X.-S., & Hossein Gandomi, A. (2012). Bat algorithm: A novel approach for global engineering optimization. *Engineering Computations, 29*, 464–483.
Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. In Nature inspired cooperative strategies for optimization (NICSO 2010) (pp. 65-74): Springer.
Yi, W., Li, X., Gao, L., Zhou, Y., & Huang, J. (2016). ε constrained differential evolution with pre-estimated comparison using gradient-based approximation for constrained optimization problems. *Expert Systems with Applications, 44*, 37–49.
Young, M. T., Hinkle, J. D., Kannan, R., & Ramanathan, A. (2020). Distributed Bayesian optimization of deep reinforcement learning algorithms. *Journal of Parallel and Distributed Computing, 139*, 43–52.
Yu, K., Wang, X., & Wang, Z. (2016). Constrained optimization based on improved teaching–learning-based optimization algorithm. *Information sciences, 352*, 61–78.
Yun, Y. (2005). Study on Adaptive Hybrid Genetic Algorithm and Its Applications to Engineering Design Problems (Unpublished Ph. D. Dissertation), Waseda University, Tokyo.
Zhang, C., Lin, Q., Gao, L., & Li, X. (2015). Backtracking Search Algorithm with three constraint handling methods for constrained optimization problems. *Expert Systems with Applications, 42*, 7831–7845.
Zhang, M., Luo, W., & Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Information sciences, 178*, 3043–3074.
Zhang, N., & Ding, S. (2017). Unsupervised and semi-supervised extreme learning machine with wavelet kernel for high dimensional data. *Memetic Computing, 9*, 129–139.
Zhang, N., Ding, S., Zhang, J., & Xue, Y. (2017). Research on point-wise gated deep networks. *Applied Soft Computing, 52*, 1210–1221.
Zhang, N., Ding, S., Zhang, J., & Xue, Y. (2018). An overview on restricted Boltzmann machines. *Neurocomputing, 275*, 1186–1199.
Zhao, J.-Q., Wang, L., Zeng, P., & Fan, W.-H. (2012). An effective hybrid genetic algorithm with flexible allowance technique for constrained engineering design optimization. *Expert Systems with Applications, 39*, 6041–6051.