Journal of Ambient Intelligence and Humanized Computing
https://doi.org/10.1007/s12652-018-0924-y

ORIGINAL RESEARCH

A hybrid crow search algorithm based on rough searching scheme for solving engineering optimization problems

Aboul Ella Hassanien[1] · Rizk M. Rizk-Allah[2] · Mohamed Elhoseny[3]

Received: 3 April 2018 / Accepted: 20 June 2018

Abstract

In this paper, a hybrid intelligent algorithm, named rough crow search algorithm (RCSA), by combining crow search algorithm (CSA) with rough searching scheme (RSS) is presented for solving engineering optimization problems. RCSA integrates the merits of the CSA and RSS to intensify the search in the promising region where the global solution resides. In terms of robustness and efficiency of the available optimization algorithms, some algorithms may not be in a position to specify the global optimal solution precisely but can rather specify them in a 'rough sense'. Thus, the main reason for incorporating the RSS is handling the impreciseness and roughness of the available information about the global optimal, particularly for the problems with high dimensionality. By upper and lower approximations of the RST, the promising region becomes under siege. Therefore this can accelerate the optimum seeking operation and achieve the global optimum with a low computational cost. The proposed RCSA algorithm is validated on 30 benchmark problems of IEEE CEC 2005, IEEE CEC 2010 and 4 engineering design problems. The obtained results by RCSA are compared with different algorithms from the literature. The comparisons demonstrate that the RCSA outperform the other algorithms for almost all benchmark problems in terms of solution quality based on the results of statistical measures and Wilcoxon signed ranks test.

Keywords Crow search algorithm · Rough set theory · Nonlinear programming problems

1 Introduction

Large-scale nonlinear programming arises in a wide variety of scientific and engineering applications including structural optimization, engineering design, very large-scale cell layout design, economics, resource allocation and many other applications (Bartholomew-Biggs 2008; Rao 2009). In most cases there are many optimization problems that involve some attributes, such as high dimensionality and multimodality, the solution of these problems are usually a complex task. Moreover, in many instances, complex optimization problems present peaks, channels and/or valleys which make traditional deterministic methods inefficient to find the global solutions.

The traditional optimization methods (TOMs) (Rao 2009) such as Newton and steepest-descent methods always rely on the restrictions of gradient information regarding the objective function and the goodness of the initial solution. These methods can perform well for small scale problems. Challenges can be appeared when coping with complex tasks that are characterized by the non-linearity, high dimensionality, multimodality, prohibited regions induced by constraints and large search areas. Handling such complex tasks using TOMs is almost impossible or requires notable computational efforts (Xiaohui et al. 2017; Rizk-Allah et al. 2018a, b).

Alternatively, the metaheuristic algorithms (MAs) have exhibited promising performance when dealing with complex tasks that are extremely nonlinear, high dimension and multimodal (Rizk-Allah 2018; Tharwat et al. 2018; Yang 2008). The MAs have some features which include the capability of searching within a wide search area for the global

Aboul Ella Hassanien, Rizk M. Rizk-Allah, Mohamed Elhoseny: Scientific Research Group in Egypt http://www.egyptscience.net.

Mohamed Elhoseny
mohamed_elhoseny@mans.edu.eg

[1] Faculty of Computers and Information, Cairo University, Cairo, Egypt

[2] Faculty of Engineering, Menoufia University, Shibīn al-Kawm, Egypt

[3] Faculty of Computers and Information, Mansoura University, Mansoura, Egypt

Published online: 25 June 2018

or near global solutions, they established based on embraces probabilistic transition rules that preserve the diversity, no reliance on the derivatives of objective function, and they are independent on the problems nature, thus they have flexibility to be applicable to a great assortment of complex tasks. However, by the means of the NFL Theorem (No Free Lunch) (Yang 2008), no meta-heuristic algorithm can be suited to deal with all optimization problems. So developing a new algorithm or modified algorithms undoubtedly is a true challenge.

Crow search algorithm (CSA) is an efficient meta-heuristic algorithm that was developed by Askarzadeh (Alireza 2016) for solving global optimization problems. It was inspired by the intelligent behavior of crows in nature. Crows are greedy birds since they follow each other to obtain better sources of food. They have very strong memory to store and retrieve food across seasons which are superior to other birds. The process of finding food source hidden by a crow is not an easy task. The crow has an intelligent ruse that is the crow tries to cheat another crow by going to another position of the environment, if it finds another one following it. The CSA has been applied to some real-world problems such as feature selection (Sayed et al. 2017), fractional optimization (Rizk-Allah et al. 2018a, b), and nonlinear optimization problems (Mohit et al. 2017). However as a new algorithm, CSA acquires some disadvantages. The first is that the updating mechanism employs unidirectional search which deteriorates the diversity of solutions and can lead to the stuck in local solution. The second is that no sieging strategy regarding the promising region is utilized and this may lead to the running without improvement in the quality of solution.

This paper presents a hybrid intelligent algorithm, named rough crow search algorithm (RCSA) for solving IEEE CEC 2005, IEEE CEC 2010 benchmark problems and engineering design problems. The proposed methodology operates in two phases: in the first phase, an improved version of the CSA is introduced based on two modifications namely, changing the crow flight length dynamically as well as searching in the opposite direction. However, the effective performance of CSA in solving optimization problems, it cannot perform well for all test problems. Thus, the second phase incorporates the RSS which is inspired by Pawlak's rough set theory to avoid the trapping in the local optimum. Meanwhile this phase breaks new regions in the search space to improve the exploration search. Furthermore, these regions are shrunken with the iteration to obtain precise optimal solution. Therefore, the embedding of the RSS phase with the CSA is a prudent way to prevent the premature convergence of the swarm and avoid the local solutions.

The rest of the paper is arranged as follows. Section 2 introduces the related work of this study. Section 3 provides the basics of crow search algorithm and rough set theory. Section 4 presents the proposed rough crow search algorithm in details. In Sect. 5, the results and discussions are provided, and finally the conclusions and future work are presented in Sect. 6.

2 Related work

This section provides the preliminaries of the nonlinear programming problem (NLPP). Also, the mechanisms of metaheuristic techniques and their challenges are discussed. Eventually, the motivation and main contribution of the proposed work are showed.

2.1 Problem formulation

A nonlinear programming problem (NLPP) is stated as follows (Rao 2009):

$$\begin{aligned} &\text{Min}_{\Omega}\, f(\mathbf{x}) = f(x_1, x_2, \dots, x_n) \\ &\text{Subject to: } \mathbf{x} \in \Omega, \end{aligned} \tag{1}$$

$$\begin{aligned} \Omega = \{\mathbf{x} |\, & g_j(\mathbf{x}) \le 0, j = 1, \dots, q, h_j(\mathbf{x}) = 0, j = q+1, \\ & \dots, m, LB_i \le x_i \le UB_i, i = 1, \dots, n\} \end{aligned} \tag{2}$$

where $f(\mathbf{x})$ is the objective function, $\mathbf{x} = (x_1, x_2, \dots, x_n)$ is a vector of n decision variables from some universe Ω, Ω contains all possible $\mathbf{x}$ that can be used to satisfy an evaluation of $f(\mathbf{x})$ and its constraints, and LB_i and UB_i represent the lower bound and the upper bound for decision variable x_i, respectively. There are q inequality constraints $g_i(\mathbf{x})$ and $(m-q)$ equality constraints $h_j(\mathbf{x})$.

The method for finding the global optimum of any function (may not be unique) is referred to as global optimization. In general, the global minimum of a single-objective problem is presented in Definition 1 (Rao 2009):

Definition 1 (*The global minimum*) Given a function $f : \Omega \subseteq \mathbb{R}^n \to R, \Omega \ne \phi$, for $\mathbf{x} \in \Omega$ the value $f^* \triangleq f(\mathbf{x}^*) > -\infty$ is called a global minimum if and only if:

$$\forall\, \mathbf{x} \in \Omega : f(\mathbf{x}^*) \le f(\mathbf{x}) \tag{3}$$

where $\mathbf{x}^*$ is by definition the global minimum solution, $f(.)$ is the objective function, $\mathbb{R}^n$ is an n-dimensional real space and the set Ω is the feasible region of $\mathbf{x}$. The goal of determining the global minimum solution is called the global optimization problem for a single-objective problem.

2.2 Literature review

Researchers have been relied on metaheuristics algorithms (Yang 2008) because of their superior abilities over the traditional optimization methods, especially for complicated optimization problems. The main reason behind their superior abilities is caused by the following features: flexibility, simplicity, derivative free approaches and ability to avoid local optima. They can be divided into two main categories: evolutionary computational algorithms (ECAs) and swarm optimization algorithms (SOAs). ECAs mimic the biological evolutionary mechanism to solve optimization problems. The most well-known paradigms of evolutionary algorithms contain genetic algorithm (GA) (Rubén et al. 2015), and differential evolution (DE) (Xiang and Wang 2015). SOAs generally imitate the collective behavior of animals such as birds, ants and bees. Particle swarm optimization (PSO) (Chijun et al. 2016), ant colony optimization (ACO) (Mousa et al. 2011) and firefly algorithm (FA) (Rizk-Allah et al. 2013) are the most famous paradigms of the SOAs. These algorithms outperform the traditional numerical methods on providing better solutions for some difficult and complicated real-world optimization problems (Rizk-Allah et al. 2017a, b, 2018a, b; Elhoseny et al. 2018a; Metawa et al. 2017).

According to NFL Theorem (No Free Lunch) (Yang 2008), no metaheuristic algorithm is suited appropriately for solving all optimization problems, where it can achieve very promising results for a set of problems, and can show poor performance in a set of different problems. Therefore the integrations with some strategies are established for obtaining effective performance. In this regard, we integrate rough set theory (RST) that was proposed by Pawlak (1982) with the crow search algorithm (CSA). RST presents an extension of the classical set theory and differ from the fuzzy set theory in its independence on any prior knowledge. It expresses the vagueness not by the means of member relationship but by employing the boundary region of a set. RST relies on replacing any vague concept, namely a subset of the universe, by two crisp concepts, which are called the upper approximation and the lower approximation of the vague concept. The upper one represents the maximal crisp set while the lower one represents the minimal crisp set of the vague concept. The boundary region is the difference between the upper and the lower approximations. Naturally, the presence of ill posed data in real-world problems is inevitable, so applying the RST methodology is a vital step. Since its inception by Pawlak (1982), it has attracted the attention of scientists and researchers in many fields such as feature selection (Shu and Shen 2014), knowledge discovery (Li et al. 2009), and attribute reduction (Xiuyi et al. 2016) and among others (Jie et al. 2017; Rizk-Allah 2016).

2.3 Motivation and contribution of the study

The theoretical researches on the optimization algorithms in the literature have been mainly concerned with two directions: improving the current techniques and hybridizing different algorithms. The goal of these directions is mainly to improve the diversity, prevent the premature convergence and increase the convergence rate. However, despite the successful test of these algorithms on some optimization problems and their high convergence speeds, theses algorithms suffer from premature convergence and weak diversity, particularly when handling highly nonlinear optimization problems and/or the optimum solution resides in a tiny subset of the search space. Further, the conflicting between precision and computing time makes these methods often yield an unsatisfactory solution that is characterized by lack of precision and slow convergence. These disadvantages are the main motivations of this work.

This paper is motivated by several features that distinguish the proposed methodology over the existing in the literatures. First, modified variant of the crow search algorithm (CSA) based on opposite direction search is introduced to preserve the exploration ability. Second, the integration between the rough searching scheme (RSS) that is inspired by Pawlak's rough set theory and crow search algorithm (CSA) to solve global optimization problems which have not been studied yet. Third, the rough searching scheme (RSS)-based evolved and shrunken regions can over overcome the drawbacks of many algorithms. Lastly, solving large scale optimization problems have not received adequate attention yet. Hence solving these problems to optimality undoubtedly becomes a true challenge.

The main contributions of this study are as follows:

1. RCSA is proposed for solving large scale optimization tasks which integrates the merits of two phases namely: crow search algorithm (CSA) and rough searching scheme (RSS).
2. CSA phase exhibits a dynamic flight length to adjust the tendency of approaching the optimal solution.
3. An opposition-based learning is adopted for updating the solution to improve the diversity of solutions.
4. RSS is proposed to siege the promising regions and then this can refine the quality of solution and avoiding the local solution.
5. The effectiveness of RCSA is investigated and validated through comprehensive experiments and comparisons for solving IEEE CEC 2005, IEEE CEC 2010 benchmark problems and engineering design problems.

3 Methods and materials

This section describes the basics of crow search algorithm (CSA) and the preliminaries of rough set theory.

3.1 Crow search algorithm (CSA)

Crow search algorithm (CSA) is a novel metaheuristic algorithm that is proposed by Askarzadeh (Alireza 2016) for solving optimization problems. It is inspired by the cleverness of crows in finding food sources. The classical CSA consists of three consecutive phases. Firstly, the position of hiding place of each crow is created randomly and the memory of each crow is initialized with this position as the best experience. Secondly, crow evaluates the quality of its position according to the objective function. Finally, crow randomly selects one of the flock crows and follows it to discover the position of the foods hidden by this crow. If the found position of the food is tasty, the crow updates its position. Otherwise, the crow stays in the current position and does not move to the generated position. The procedure of the CSA is summarized as follows (Alireza 2016).

Step 1: Initialize a swarm of crows within the n dimensional search space, where the algorithm assigns a random vector $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,n})$ for the ith crow, $i = 1, 2, \dots, N$. Furthermore, each crow of the swarm is characterized by its memory (i.e., initially, the memory of each crow is filled with the initial position, $\mathbf{m}_i = (m_{i,1}, m_{i,2}, \dots, m_{i,n})$, where the crows have no experience about the food sources).

Step 2: Each crow is evaluated according to the quality of its position which is related to the desired objective function.

Step 3: Crows create new positions in the search space as follows: crow i selects one of the flock crows randomly, i.e., crow j, and follows it to discover the position of the foods hidden by this crow, where the new position of crow i is generated as follows:

$$x_{i,Iter+1} = \begin{cases} x_{i,Iter} + r_i \times fl_{i,Iter}\,(m_{j,Iter} - x_{i,Iter}) & a_j \geq AP_{j,t} \\ \text{a random position} & \text{otherwise} \end{cases} \tag{4}$$

where r_i, a_j are random numbers with uniform distribution between 0 and 1, $AP_{j,t}$ denotes the awareness probability of crow j at iteration $Iter$ and $fl_{i,t}$ denotes the flight length of crow i at iteration $Iter$. $m_{j,Iter}$ denotes the memory of crow j at iteration $Iter$.

Figure 1 shows the effect of the parameter fl on the search capability where the small values of $fl (fl \leq 1)$ leads to explore new position of crow lies on the dashed line between $m_{j,t}$ and $x_{i,t}$ as in Fig. 1a, while Fig. 1b shows that, if the value of fl is selected more than 1, the new position of crow lies on the dashed line which outside the line segment between $m_{j,t}$ and $x_{i,t}$.

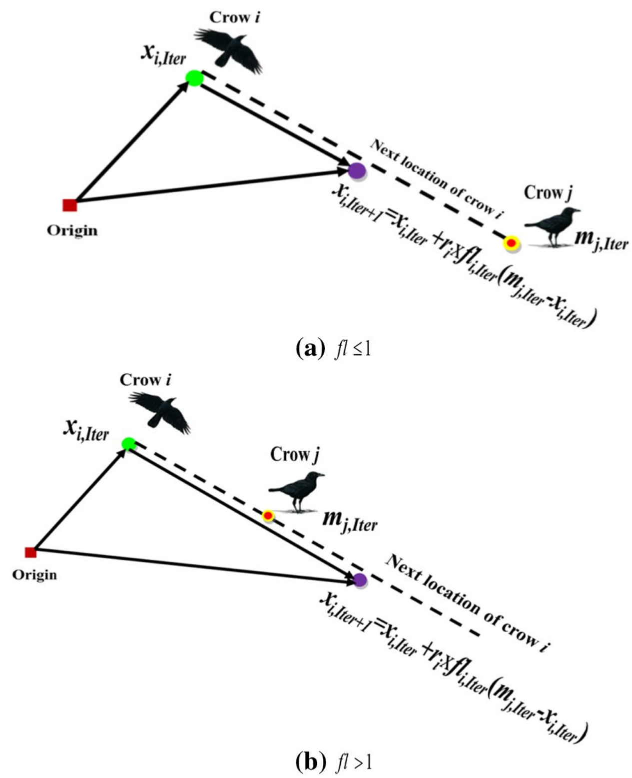


Fig. 1 Searching mechanism by the crow in the two states: **a** $fl \leq 1$ and **b** $fl > 1$

Step 4: After generating the crow's positions, the new positions are evaluated and each crow updates its memory as follows:

$$m_{i,Iter+1} = \begin{cases} x_{i,Iter+1} & f(x_{i,Iter+1}) \succ f(m_{i,Iter}) \\ m_{i,Iter} & \text{otherwise} \end{cases} \tag{5}$$

where $f(.)$ denotes the objective function value, $\succ$ denotes better than.

The nature behavior of crow is characterized by memorizing the position of hidden places of food and retrieving it across seasons. In this regard, it is assumed that each crow memorizes the position of hidden places in a memory denoted by m, thus at iteration $Iter$, the position of hidden place of crow j is denoted by $m_{j,Iter}$. In the initialize step, the memory $m_{j,Iter}$ of crow j is initialized with its initial position $x_{j,Iter}$, then this memory is updated at each iteration by Eq. (5) to attain best position of food source (hidden place). Equation (5) operates by filling the memory of the crow with its new position if it is better that than the sored one.

Step 5: End the algorithm if the maximum number of generations is met and the best position of the memory in terms of the objective function value is reported as the solution of the optimization problem; otherwise, go back to Step 3.

3.2 Rough set theory (RST)

The basic concept of the RST is the indiscernibility relation, which is generated by the information about the objects (Pawlak 1982). Because the discerning knowledge is lack, one cannot identify some objects based on the available information. The indiscernibility relation expresses this fact by considering granules of indiscernible objects as a fundamental basis. Some relevant concepts of the RST are presented in Pawlak (1982) (i.e., see Appendix 1).

4 The proposed RCSA algorithm

The metaheuristic algorithms have been devised to overcome the computational drawbacks of existing numerical algorithms such as complex derivatives, sensitivity to initial values and the large amount of enumeration memory required. Their conventional procedures for finding the optimal solution are iterative based and depend on randomness to imitate natural phenomena. Thus the process for searching the global optimal solution would reveal that some of metaheuristic algorithms fail to find a precise value for the global optimal solution but they can obtain an approximate value or rough sense value. In such situations, it is desirable to provide more exploration in finding the global solution and to prevent premature convergence of the swarm. Towards this objective, we focused in this study on the hybridization between RSS and CSA. This hybridization is called rough crow search algorithm (RCSA). The proposed RCSA operates in two phases: in the first one, CSA is implemented as global optimization system to find an approximate solution of the global optimization problem. In the second phase, RSS is introduced to improve the solution quality through the roughness of the obtained optimal solution so far. By this way, the roughness of the obtained optimal solution can be represented as a pair of precise concepts based on the lower and upper approximations which are used to constitute the interval of boundary region. After that, new solutions are randomly generated inside this region to enhance the diversity of solutions and achieve an effective exploration to avoid premature convergence of the swarm. We start the explanation of the RCSA as follows.

4.1 Algorithm initialization

Instead of fixing the parameters, the fine-tuning process of the algorithm parameters may have major influence in faster convergence and final outcome. As a first improvement, RCSA introduces a new modification on the flight length parameter such that changes dynamically with iteration number, instead of the fixed value. The second improvement has been considered the searching in the opposite directions. In addition, RSS is introduced to define the bounds for the obtained optimal solution so far and then new solutions are generated randomly inside these bounds.

4.2 Rough CSA

Phase 1: CSA

In CSA, crucial influence on algorithm performance refers to the calculation of the hiding places of a crow. Basic implementation of this metaheuristic technique assumes a fixed value of the flight length which cannot be changed during iterations. The main drawback of this technique appears in the flight length value that the algorithm needs to cover the overall search space for finding the optimal solution. The small value of the flight length explores the solutions inside the line segment and the large value of the flight length explores the solutions outside the line segment. Thus, this is often not a good choice, especially when dealing with more complex nonlinear and multimodal problems. In order to accelerate the convergence, eliminate the drawbacks which caused by fixed values of the flight length and balance exploration and exploitation, the flight length is changed dynamically with iteration number as shown in Fig. 2 using the following equation (Pan et al. 2014):

$$fl_{Iter} = fl_{\max} \cdot \exp\left(\log\left(\frac{fl_{\min}}{fl_{\max}}\right) \cdot \left(\frac{Iter}{Iter_{\max}}\right)\right) \tag{6}$$

where fl_{Iter} is the flight length in each iteration, $fl_{\min}$ is the minimum flight length, $fl_{\max}$ is the maximum flight length, $Iter$ is the iteration number and $Iter_{\max}$ is the maximum iteration number.

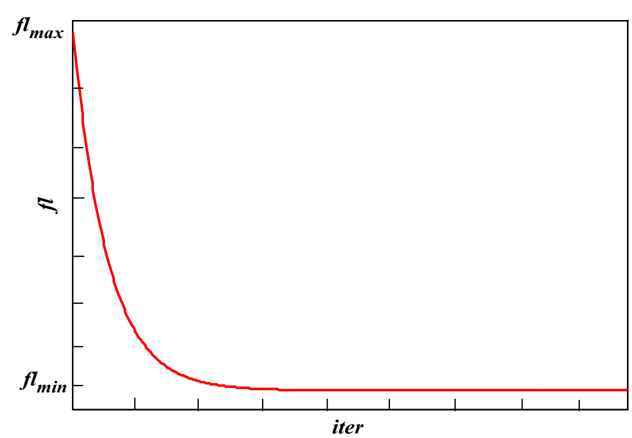


Fig. 2 Representation of fl versus iterations

In addition, we modified the Eq. (4) so that it involves the new positions inside and outside the segment that is extended equally on both sides determined by a specified value for $fl_{\max}$.

$$\mathbf{x}_{i,Iter+1} = \begin{cases} \mathbf{x}_{i,Iter} + r_i \times fl_{i,Iter} \left(m_{j,Iter} - \mathbf{x}_{i,Iter}\right) & \text{if } a_j \geq AP_{j,Iter} \\ \mathbf{x}_{i,Iter} - r_i \times fl_{i,Iter} \left(m_{j,Iter} - \mathbf{x}_{i,Iter}\right) & \text{else } d = i \\ LB + rand \times (UB - LB) & \text{otherwise} \end{cases}, \quad i = 1, 2, \ldots, N \tag{7}$$

where $d \in \{1, 2, \ldots, N\}$ is a randomly chosen index.

To further enhance the intensive search, we force the crows to search towards the opposite directions (Seif and Ahmadi 2015) in the random manner through use the negative sign that is inspired by fooling process. Equation (7) presents a new modification on the Eq. (4), meanwhile Eq. (7) is introduced with the aim of generating a new position of crow and its opposite instead of the generating a position only as in Eq. (4). In this context, the flight length, fl_{Iter}, that is introduced in Eq. (7) is evolved by Eq. (6) instead of using fixed value as in traditional CSA.

Phase 2: Rough searching scheme (RSS)

However, the optimization producers of the CSA phase yields an approximated optimal solution $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ which is not specified in precise crisp term, rough searching scheme-based local search is proposed in this paper to guide CSA for approaching the global optimal solution, where the approximated optimal solution is converted into rough number. Afterwards the rough interval is obtained through the upper and lower approximations for each variable. Therefore a new offspring is generated inside the obtained rough interval. The detailed description of this phase is described as follows:

Step 1: Information system

This step uses the swarm solutions as a key function for implementing the information system. The information system is denoted as the ordered pair (U, A), where each individual is treated as an object (solution) of a non-empty finite set U. Attribute set $A = \{d_1, d_2, \ldots, d_n\}$ is a non-empty finite set of attributes (i.e., the dimensions of the candidate problem, where each dimension is represented as a conditional attribute).

Step 2: Rough approximations

The ordered pair $S = (U, C)$ is called an approximation space generated by C on U, where U is a non-empty finite set of solutions per dimension (i.e., obtained solutions from CSA phase where each solution is represented as a class) and C is a reflexive relation on Uthat partitions U into N classes, i.e., $U/C = \{\{x_1\}, \{x_2\}, \ldots, \{x_i\}, \ldots, \{x_N\}\}$, where $\{x_i\}$ is the ith class and the all classes are expressed in increasing ordered such that $\{x_1\} \leq \{x_2\} \leq \cdots \leq \{x_N\}$. Additionally, $\{x_i\} \leq \{x_j\}$ if and only if $x_i \leq x_j$. For each dimension d_j, $j = 1, 2, \ldots, n$, the lower approximation of x_i, $i = 1, 2, \ldots, N$, is denoted as $\underline{Apr}(x_i)$ and can be defined as follows:

$$\underline{Apr}(x_i) = \cup\{y \in U/C(y) \leq x_i\}. \tag{8}$$

The upper approximation $\overline{Apr}(x_i)$ of x_i can be defined as follows

$$\overline{Apr}(x_i) = \cup\{y \in U/C(y) \geq x_i\}. \tag{9}$$

Accordingly, the boundary region of x_i is given by

$$\begin{aligned} BN(x_i) &= \cup\{y \in U/C(y) \neq x_i\} \\ &= \{y \in U/C(y) > x_i\} \cup \{y \in U/C(y) < x_i\}. \end{aligned} \tag{10}$$

In classical RST, any subset $X \subseteq U$ is described by its lower and upper approximations, i.e., $\underline{BX} \subseteq X \subseteq \overline{BX}$. As a counterpart when dealing with crisp value, the less than or equal ($\leq$) and the greater than or equal ($\geq$) are used instead of using the subset ($\subseteq$) and superset ($\supseteq$), respectively.

For example, if we have $U/C = \{\{4\}, \{5\}, \{7\}\}$, then the lower and upper approximations for each class using Eqs. 8 and 9 can be obtained as follows:

$$\begin{array}{lll} \underline{Apr}(4) = 4 & \overline{Apr}(4) = \{4 + 5 + 7\} & BN(4) = \{5, 7\} \\ \underline{Apr}(5) = \{4, 5\} & \overline{Apr}(5) = \{5, 7\} & BN(5) = \{4, 7\}. \\ \underline{Apr}(7) = \{4, 5, 7\} & \overline{Apr}(7) = 7 & BN(7) = \{4, 5\} \end{array}$$

The Step 2 (i.e., of rough approximations) makes usage of the upper and lower approximations to describe the proposed process. It is extended from the basics of the rough set theory that are described in Sect. 2.3. Hence the obtained solutions for ith dimension are represented as the degrees under increasing order condition (i.e., $x_1 < x_2 < \cdots < x_N$), then the subset ($\subseteq$) and superset ($\supseteq$) are analogous to the less than or equal ($\leq$) and the greater than or equal ($\geq$), respectively. In simple words, for a degree, x_i, in a set of ordered degrees, the lower approximation of x_i contains all the solutions (degrees) in the information system that have values equal to or less than x_i. The upper approximation of x_i contains all the solutions (degrees) in the same information system that have values equal to or greater than x_i; and the boundary region of x_i contains all the degrees in the information table that have different values from x_i.

Step 3: Rough interval

Based on the approximations of a class defined above, the so-called rough number can be defined as follows: the degree, x_i, of the ith dimension can be represented by its rough number composed of the lower bound (x_i^{LB}) and the upper bound (x_i^{UB}) is denoted by $RN(x_i)$. Mathematically,

$$x_i^{LB} = \frac{1}{N_{LB}} \sum y \mid y \in \underline{Apr}(x_i) \tag{11}$$

$$x_i^{UB} = \frac{1}{N_{UB}} \sum y \mid y \in \overline{Apr}(x_i) \tag{12}$$

where N_{LB} is the number of objects contained in the lower approximation of x_i and N_{UB} is the number of objects contained in the upper approximation of x_i.

The interval between the lower bound (x_i^{LB}) and the upper bound (x_i^{UB}) is known as the rough boundary interval, which is denoted as $RBI(x_i)$ as follows:

$$RBI(x_i) = x_i^{UB} - x_i^{LB}. \tag{13}$$

Definition 6 Any vague class is characterized by a so-called rough number ($RN(x_i)$) consisting of the lower bound and the upper bound of the said class as follows:

$$RN(x_i) = [x_i^{LB}, x_i^{UB}]. \tag{14}$$

Remark According to the above example, the Eqs. 11 and 12 can be calculated as follows:

$$\begin{array}{lll} x_1^{LB} = 4 & x_1^{UB} = 5.33, & RN(x_1) = [4, 5.33] \\ x_2^{LB} = 4.5 & x_2^{UB} = 6, & RN(x_2) = [4.5, 6] \\ x_3^{LB} = 5.33 & x_3^{UB} = 7, & RN(x_3) = [5.33, 7] \end{array}.$$

By using Eq. (14) we can generate new solutions randomly inside each interval and the unified interval. The unified interval of ith dimension is computed as follows.

The rough solution of ith dimension is computed as follows:

$$x_i^{LB} = (x_{i1}^{LB} + x_{i2}^{LB} + \cdots + x_{iN}^{LB})/N \tag{15}$$

$$x_i^{UB} = (x_{i1}^{UB} + x_{i2}^{UB} + \cdots + x_{iN}^{UB})/N. \tag{16}$$

Briefly, the lower bound of a ith dimension is the mean value of the degrees contained in its lower approximations whereas the upper bound of a ith dimension is the mean value of the degrees contained in its upper approximations. The rough boundary interval of a ith dimension is the difference between its upper and lower bounds, which describes the vagueness of the said class. A class with a larger rough boundary interval is said to be vague, or less precise.

Therefore, the unified rough intervals or the regions of interest for overall dimensions are represented as follows:

$$RI(\mathbf{x}) = \{[x_1^{LB}, x_1^{UB}], [x_2^{LB}, x_2^{UB}], \ldots, [x_n^{LB}, x_n^{UB}]\}. \tag{17}$$

Definition 7 The optimal solution is a rough number denoted by $\mathbf{x}^*$, whose lower and upper bounds are denoted by $\mathbf{x}^{UB}$ and $\mathbf{x}^{LB}$, respectively.

Remark 1 If $\mathbf{x}^{UB} = \mathbf{x}^{LB}$, then the optimal solution $\mathbf{x}^*$ is exact (crisp), otherwise $\mathbf{x}^*$ is inexact (rough).

Step 4: Generation

In this step, new solutions are generated inside the new intervals randomly, and then these solutions are evolved through the use of the RCSA phase.

Step 5: Evaluation

In this step, the solutions are evaluated to judge if the best fitness is superior to the previous one, if so, update the best fitness value and at this moment, the best location is updated.

Step 6: Stopping criterion

This phase is terminated when either the maximum number of generations has been produced or the obtained optimal solution is exact (crisp). Schematic diagram of the rough set phase is demonstrated in Fig. 3.

In summary, the first phase of the proposed, CSA, is responsible for delivering a population of solutions for the second phase, RSS, where each dimension is represented by a vector of N-degrees. To effectively create a reliable regions in the search space, two concepts are introduced, namely rough set approximations and rough interval with the aim to achieve a prices optimal solution. The rough set approximations consist of lower and upper approximations, where the lower approximation for certain degree is defined by all degrees that are less or equal this degree while the upper approximation for certain degree is defined by all degrees that are greater or equal this degree. Afterwards, the concept of rough interval in carried out based on these approximations, where the rough interval is represented by the lower bound and upper bound. The lower (upper) bound for any degree is the mean of all degrees in its lower (upper) approximation. By the two concepts, the search is concentrated in the reliable region and therefore this can achieve more accurate solutions as well as save the computational time.

Figure 4 shows the general architecture of the proposed RCSA algorithm, where the highlighted boxes represent the introduced modifications on the original one. Figure 4 starts with initial positions associated with initial

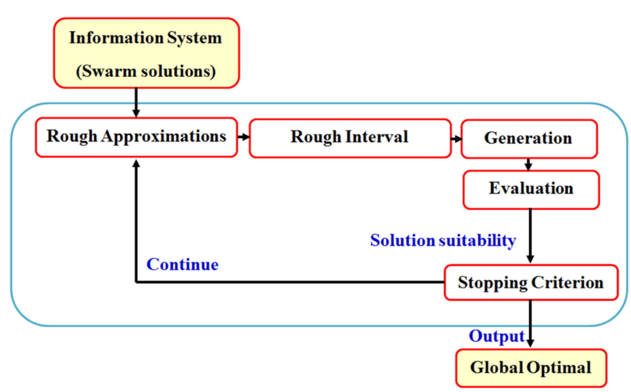


Fig. 3 Schematic diagram of rough searching scheme phase

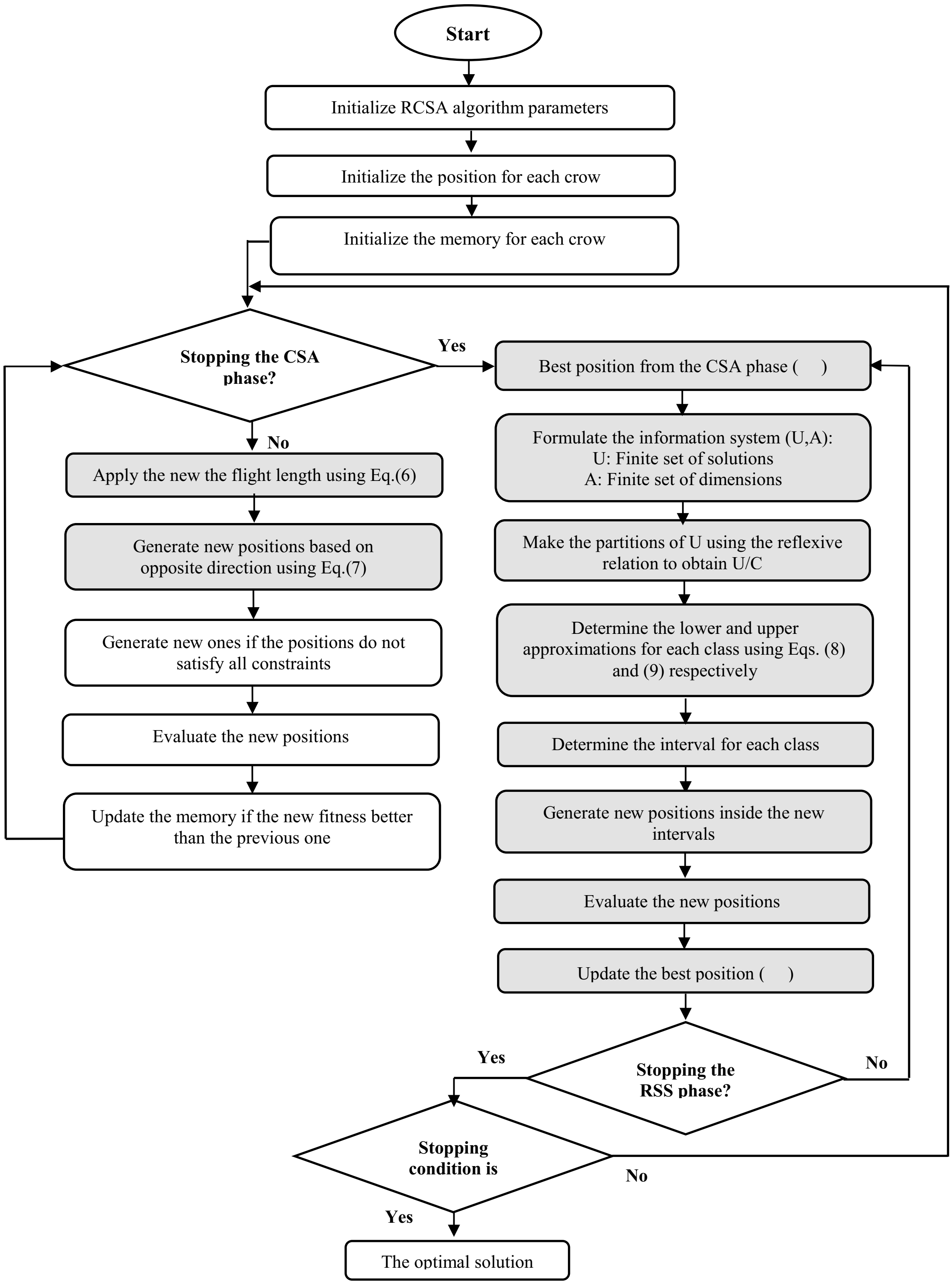


Fig. 4 Architecture of the proposed RCSA algorithm

CSA phase

Input: Parameters: N, $Iter_{\max}$, AP, $fl_{\max}$ and $fl_{\min}$.

Initialize the positions of crows randomly in the search space

Evaluate the initial positions

Initialize the memory of crows with the Initial positions

// Looping

while $Iter <= Iter_{\max}$ **do**

$$fl_{Iter} = fl_{\max}.\exp\left(\log\left(\frac{fl_{\min}}{fl_{\max}}\right).\left(\frac{Iter}{Iter_{\max}}\right)\right)$$

for $i = 1\ to\ N$

$d =$ a random integer in the range of $[1, N]$

$$x_{i,Iter+1} = \begin{cases} x_{i,Iter} + r_i \times fl_{i,Iter}(m_{j,Iter} - x_{i,Iter}) & \text{if } a_j \geq AP_{j,Iter} \\ x_{i,Iter} - r_i \times fl_{i,Iter}(m_{j,Iter} - x_{i,Iter}) & \text{else } d = i \\ LB + rand.(UB - LB) & \text{otherwise} \end{cases}, i = 1, 2, ..., N$$

if $x_{i,Iter+1} > UB$, **then** $x_{i,Iter+1} = UB$,

if $x_{i,Iter+1} < LB$, **then** $x_{i,Iter+1} = LB$

end for

Evaluate the obtained positions of the crows

Update the memory of crows

End while

$\mathbf{x}^* =$ Best position from the CSA phase

RSS phase

Formulate the information system

Rank the obtained positions from the CSA phase

Calculate the rough interval for ith dimension as follows

$$x_i^{LB} = (x_i^{1LB}, x_i^{2LB}, ..., x_i^{NLB})/N \ \ \& \ \ x_i^{UB} = (x_i^{1UB}, x_i^{2UB}, ..., x_i^{NUB})/N$$

Repeat

Generate new positions randomly

for $i = 1\ to\ N$

$$\mathbf{x}'_i = \begin{cases} \mathbf{x}^{LB} + rand.(\mathbf{x}^* - \mathbf{x}^{LB}) & if\ rand \geq 0.5 \\ \mathbf{x}^* + rand.(\mathbf{x}^{UB} - \mathbf{x}^*) & if\ rand < 0.5 \end{cases}$$

End for

Evaluate the obtained positions

Update the best position

Until $Iter = Iter_{\max}$

end Looping

Output: $\mathbf{x}^*$

Fig. 5 The pseudo code of the proposed RCSA algorithm memory then the updating mechanism of these positions and memory is performed according to Eqs. (7) and (5) respectively until the stopping condition of CSA phase is satisfied. Afterwards the rough searching scheme (RSS) operates until its stopping condition is satisfied where RSS receives the solutions from the CSA phase with the aim to construct the rough interval by the means of the upper and lower approximations for each variable. Therefore a new offspring is generated inside the obtained rough interval and the best one among CSA and RSS phases is survival. Also pseudo code of the proposed RCSA can be summarized as shown in Fig. 5.

5 Experiments and results

5.1 Test functions and parameter settings

In this section, the performance of the proposed RCSA algorithm is tested on 25 CEC'2005 benchmark functions (Suganthan et al. 2005; Sedlaczek and Eberhard 2005). The mathematical description these functions, types of functions and number of dimensions (Suganthan et al. 2005) are given in Table 1. The robustness and effectiveness of the proposed RCSA are validated through comparing it with the prominent algorithms from literature. The algorithm is coded in MATLAB 7, running on a computer with an Intel Core I 5 (1.8 GHz) processor and 4 GB RAM memory.

Additionally, the parameter configurations of the RCSA and CSA algorithms including population size, the number of iterations and awareness probability are based on the suggestions in the corresponding literature (Alireza 2016) while the flight length is introduced by a new manner to change dynamically [i.e., see Eq. (6)] where the maximum flight length is considered as the maximum radius of search and minimum flight length is considered as the minimum radius (i.e., see Table 2).

Table 2 Parameter settings of RCSA

Population size	50
The number of iterations	500
Awareness probability (*AP*)	0.1
Maximum flight length ($fl_{\max}$)	$(UB - LB)/2$
Minimum flight length ($fl_{\min}$)	10^{-5}
Flight length (*fl*)	Changes dynamically

5.2 Performance analysis using the statistical measures

To completely evaluate the performance of the proposed RCSA algorithm, the comparison between the global solution and the obtained solution by the RCSA for each test function is reported as in Table 3 where in each tested case, the solution is demonstrated before and after incorporating the RSS phase. From the Table 3, we can note that the obtained solutions after incorporating the RSS phase are more accurate and converge to the optimal value solutions than that obtained without incorporating the RSS phase.

As indicated in Table 3, the proposed RCSA algorithm gives the exact optimum results for the test functions 1–12 when the algorithm is implemented with and without RSS phase, while RCSA algorithm performs better results than the algorithm without RSS phase for the test functions 13–25.

Beside the comparison with global optimal solution, we additionally use the statistical measures (i.e., see Table 4) such as best, mean, median and worst objective values as well as their standard deviations and average time are obtained over 50 independent runs for each test problem.

Table 1 Benchmark functions

ID	Function name	D	C	ID	Function name	D	C
F_1	Shifted Sphere Function	10	U	F_9	Shifted Rastrigin's Function	10	M
F_2	Shifted Schwefel's Problem 1.2	10	U	F_{10}	Shifted Rotated Rastrigin's Function	10	M
F_3	Shifted Rotated High Conditioned Elliptic Function	10	U	F_{11}	Shifted Rotated Weierstrass Function	10	M
F_4	Shifted Schwefel's Problem 1.2 with Noise in Fitness	10	U	F_{12}	Schwefel's Problem 2.13	10	M
F_5	Schwefel's Problem 2.6 with Global Optimum on Bounds	10	U	F_{13}	Expanded Extended Griewank's plus Rosenbrock's Function (F8F2)	10	M
F_6	Shifted Rosenbrock's Function	10	M	F_{14}	Shifted Rotated Expanded Scaffers F6	10	M
F_7	Shifted Rotated Griewank Function without Bounds	10	M	$F_{15}:F_{25}$	Hybrid functions: where each on has been composed from the previous functions (different in each case)	10	M
F_8	Shifted Rotated Ackley's Function with Global Optimum on Bounds	10	M				

C characteristic, *U* unimodal, *M* multimodal, *D* dimension of *n*

Table 3 Comparison between the global solution and RCSA solution without and with RSS phase

Function	Global solution	The RCSA algorithm without RSS phase	The RCSA algorithm with RSS phase	Function	Global solution	The RCSA algorithm without RSS phase	The RCSA algorithm with RSS phase
F_1	−450.0000	−449.5842	−450.0000	F_{14}	−300.0000	−299.7108	−299.9610
F_2	−450.0000	−450.0000	−450.0000	F_{15}	120.0000	120.0000	120.0000
F_3	−450.0000	−450.0000	−450.0000	F_{16}	120.0000	122.9352	122.9287
F_4	−450.0000	−450.0000	−450.0000	F_{17}	120.0000	128.0572	121.0399
F_5	−310.0000	−309.9890	−310.0000	F_{18}	10.0000	300.0409	300.0342
F_6	390.0000	390.0000	390.0000	F_{19}	10.0000	300.1341	300.1185
F_7	−180.0000	−179.9994	−180.0000	F_{20}	10.0000	300.3235	300.3200
F_8	−140.0000	−139.99952	−140.0000	F_{21}	360.0000	500.0374	500.0278
F_9	−330.0000	−330.0000	−330.0000	F_{22}	360.0000	700.8376	532.1342
F_{10}	−330.0000	−330.0000	−330.0000	F_{23}	360.0000	463.0758	463.0758
F_{11}	90.0000	90.0000	90.0000	F_{24}	260.0000	293.7459	265.4191
F_{12}	−460.0000	−460.0000	−460.0000	F_{25}	260.0000	394.8046	375.7108
F_{13}	−130.0000	−129.6139	−129.9901				

Table 4 Statistical results of RCSA for the overall test functions

Function	Best	Mean	Median	Worst	SD	Ave. time (s)
F_1	−450.0000	−449.9974	−450.0000	−449.9574	8.1340E−3	0.6928095
F_2	−450.0000	−450.0000	−450.0000	−449.99912	1.69680E−4	0.8602075
F_3	−450.0000	−449.8758	−449.9964	−448.2176	3.6265E−1	1.7043
F_4	−450.0000	−450.0000	−450.0000	−450.0000	1.5951E−5	0.9525
F_5	−310.0000	−309.9935	−309.8324	−306.2820	8.2210E−1	1.7133
F_6	390.0000	−399.4495	−389.1946	−384.1468	1.4290	0.9599
F_7	−180.0000	−180.0000	−180.0000	−179.9279	1.4658E−2	0.8569
F_8	−140.0000	−140.0000	−140.0000	−139.9978	4.2917E−4	0.5137
F_9	−330.0000	−330.0000	−330.0000	−330.0000	0.0000	0.5671
F_{10}	−330.0000	−330.0000	−330.0000	−330.0000	0.0000	0.8693
F_{11}	90.0000	90.0000	90.0000	90.0000	0.0000	0.5613
F_{12}	−460.0000	−460.0000	−460.0000	−460.0000	0.0000	0.6504
F_{13}	−129.9901	−129.7794	−129.7607	−129.6294	8.9215E−2	0.7215
F_{14}	−299.9610	−299.9289	−299.9461	−299.8408	3.9559E−2	1.2306
F_{15}	120.0000	120.0000	120.0000	120.0000	0.0000	0.9582
F_{16}	122.9287	122.9287	122.9287	122.9287	2.9032E−14	0.9965
F_{17}	121.0399	121.1665	121.0399	128.1268	9.4701E−1	1.1208
F_{18}	300.0342	300.0423	300.0423	300.1388	1.9347E−2	0.8273
F_{19}	300.1185	300.1185	300.1185	300.1185	0.0000	0.6134
F_{20}	300.3200	300.3209	300.32008	300.3235	1.7073E−3	1.2885
F_{21}	500.0278	5017.2094	500.0707	5039.4986	2.0850	1.8769
F_{22}	532.1342	580.0254	532.1342	651.8622	60.7131	1.8652
F_{23}	463.0758	463.0758	463.0758	463.0758	6.9618E−014	0.6875
F_{24}	265.4191	269.8181	265.4191	298.5889	10.5783	0.9856
F_{25}	375.7108	383.7492	384.0652	391.2666	1.9513	1.5698

Table 5 Comparison of RCSA with other recent algorithms (best results are given in bold)

Function	PSO	IPOP-CMA-ES	CHC	SSGA	SS-BLX	SS-Arit	DE-Bin	DE-Exp	SaDE	Proposed RCSA
F_1	1.234E−4	**0.000**	2.464	8.420E−9	3.402E+1	1.064	7.716E−9	8.260E−9	8.416E−9	**0.000**
	Rank 6	Rank 1	Rank 8	Rank 5	Rank 9	Rank 7	Rank 2	Rank 3	Rank 4	Rank 1
F_2	2.595E−2	**0.000**	1.180E−2	8.719E−5	1.730	5.282	8.342E−9	8.181E−9	8.208E−9	**0.000**
	Rank 7	Rank 1	Rank 6	Rank 5	Rank 8	Rank 9	Rank 4	Rank 2	Rank 3	Rank 1
F_3	5.174E+4	**0.000**	2.699E+5	7.948E+4	1.844E+5	2.535E+5	4.233E+1	9.935E+1	6.560E++3	**0.000**
	Rank 5	Rank 1	Rank 9	Rank 6	Rank 7	Rank 8	Rank 2	Rank 3	Rank 4	Rank 1
F_4	2.488	2.932E+3	9.190E+1	2.585E−3	6.228	5.755	7.686E−9	8.350E−9	8.087E−9	**0.000**
	Rank 6	Rank 10	Rank 9	Rank 5	Rank 8	Rank 7	Rank 2	Rank 4	Rank 3	Rank 1
F_5	4.095E+2	8.104E−10	2.641E+2	1.343E+2	2.185	1.443E+1	8.608E−9	8.514E−9	8.640E−9	**0.000**
	Rank 10	Rank 2	Rank 9	Rank 8	Rank 7	Rank 6	Rank 4	Rank 3	Rank 5	Rank 1
F_6	7.310E+2	**0.000**	1.416E+6	6.171	1.145E+2	4.945E+2	7.956E−9	8.391E−9	1.612E−2	**0.000**
	Rank 8	Rank 1	Rank 9	Rank 5	Rank 6	Rank 7	Rank 2	Rank 3	Rank 4	Rank 1
F_7	2.678E+1	1.267E+3	1.269E+3	1.271E+3	1.966E+3	1.908E+3	1.266E+3	1.265E+3	1.263E+3	**0.000**
	Rank 2	Rank 6	Rank 7	Rank 8	Rank 10	Rank 9	Rank 5	Rank 4	Rank 3	Rank 1
F_8	2.043E+1	2.001E+1	2.034E+1	2.037E+1	2.035E+1	2.036E+1	2.033E+1	2.038E+1	2.032E+1	**0.000**
	Rank 10	Rank 2	Rank 5	Rank 8	Rank 6	Rank 7	Rank 4	Rank 9	Rank 3	Rank 1
F_9	1.438E+1	2.841E+1	5.886	7.286E−9	4.195	5.960	4.546	8.151E−9	8.330E−9	**0.000**
	Rank 9	Rank 10	Rank 7	Rank 2	Rank 5	Rank 8	Rank 6	Rank 3	Rank 4	Rank 1
F_{10}	1.404E+1	2.327E+1	7.123	1.712E+1	1.239E+1	2.179E+1	1.228E+1	1.118E+1	1.548E+1	**0.000**
	Rank 6	Rank 10	Rank 2	Rank 8	Rank 5	Rank 9	Rank 4	Rank 3	Rank 7	Rank 1
F_{11}	5.590	1.343	1.599	3.255	2.929	2.858	2.434	2.067	6.796	**0.000**
	Rank 9	Rank 2	Rank 3	Rank 8	Rank 7	Rank 6	Rank 5	Rank 4	Rank 10	Rank 1
F_{12}	6.362E+2	2.127E+2	7.062E+2	2.794E+2	1.506E+2	2.411E+2	1.061E+2	6.309E+1	5.634E+1	**0.000**
	Rank 9	Rank 6	Rank 10	Rank 8	Rank 5	Rank 7	Rank 4	Rank 3	Rank 2	Rank 1
F_{13}	1.503	1.134	8.297E+1	6.713E+1	3.245E+1	5.479E+1	1.573	6.403E+1	7.070E+1	**0.0099**
	Rank 3	Rank 2	Rank 10	Rank 8	Rank 5	Rank 6	Rank 4	Rank 7	Rank 9	Rank 1
F_{14}	3.304	3.775	2.073	2.264	2.796	2.970	3.073	3.158	3.415	**0.0390**
	Rank 8	Rank 10	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 9	Rank 1
F_{15}	3.398E+2	1.934E+2	2.751E+2	2.920E+2	1.136E+2	1.288E+2	3.722E+2	2.940E+2	8.423E+1	**0.000**
	Rank 9	Rank 5	Rank 6	Rank 7	Rank 3	Rank 4	Rank 10	Rank 8	Rank 2	Rank 1
F_{16}	1.333E+2	1.170E+2	9.729E+1	1.053E+2	1.041E+2	1.134E+2	1.117E+2	1.125E+2	1.227E+2	**2.9287**
	Rank 10	Rank 8	Rank 2	Rank 4	Rank 3	Rank 7	Rank 5	Rank 6	Rank 9	Rank 1
F_{17}	1.497E+2	3.389E+2	1.045E+2	1.185E+2	1.183E+2	1.279E+2	1.421E+2	1.312E+2	1.387E+2	**1.0399**
	Rank 9	Rank 10	Rank 2	Rank 4	Rank 3	Rank 5	Rank 8	Rank 6	Rank 7	Rank 1
F_{18}	8.512E+2	5.570E+2	8.799E+2	8.063E+2	7.668E+2	6.578E+2	5.097E+2	4.482E+2	5.320E+2	**2.9003E+2**
	Rank 9	Rank 5	Rank 10	Rank 8	Rank 7	Rank 6	Rank 3	Rank 2	Rank 4	Rank 1
F_{19}	8.497E+2	5.292E+2	8.798E+2	8.899E+2	7.555E+2	7.010E+2	5.012E+2	4.341E+2	5.195E+2	**2.9011E+2**
	Rank 8	Rank 5	Rank 9	Rank 10	Rank 7	Rank 6	Rank 3	Rank 2	Rank 4	Rank 1
F_{20}	8.509E+2	5.264E+2	8.960E+2	8.893E+2	7.463E+2	6.411E+2	4.928E+2	4.188E+2	4.767E+2	**2.9032E+2**
	Rank 8	Rank 5	Rank 10	Rank 9	Rank 7	Rank 6	Rank 4	Rank 2	Rank 3	Rank 1
F_{21}	9.138E+2	4.420E+2	8.158E+2	8.522E+2	4.851E+2	5.005E+2	5.240E+2	5.420E+2	5.140E+2	**1.4002E+2**
	Rank 10	Rank 2	Rank 8	Rank 9	Rank 3	Rank 4	Rank 6	Rank 7	Rank 5	Rank 1
F_{22}	8.071E+2	7.647E+2	7.742E+2	7.519E+2	6.828E+2	6.941E+2	7.715E+2	7.720E+2	7.655E+2	**1.7213E+2**
	Rank 10	Rank 5	Rank 9	Rank 4	Rank 2	Rank 3	Rank 7	Rank 8	Rank 6	Rank 1
F_{23}	1.028E+3	8.539E+2	1.075E+3	1.004E+3	5.740E+2	5.828E+2	6.337E+2	5.824E+2	6.509E+2	**1.0307E+2**
	Rank 9	Rank 7	Rank 10	Rank 8	Rank 2	Rank 4	Rank 5	Rank 3	Rank 6	Rank 1
F_{24}	4.120E+2	6.101E+2	2.959E+2	2.360E+2	2.513E+2	2.011E+2	2.060E+2	2.020E+2	2.000E+2	**0.5419E+1**
	Rank 9	Rank 10	Rank 8	Rank 6	Rank 7	Rank 3	Rank 5	Rank 4	Rank 2	Rank 1

Table 5 (continued)

Function	PSO	IPOP-CMA-ES	CHC	SSGA	SS-BLX	SS-Arit	DE-Bin	DE-Exp	SaDE	Proposed RCSA
F_{25}	5.099E+2 Rank 2	1.818E+3 Rank 10	1.764E+3 Rank 7	1.747E+3 Rank 6	1.794E+3 Rank 8	1.804E+3 Rank 9	1.744E+3 Rank 5	1.742E+3 Rank 4	1.738E+3 Rank 3	**1.1571E+2** Rank 1

Table 6 Saving of fitness for the CEC 2005 test problems

Function	Saving rate %	Function	Saving rate %
F_1	0.092485	F_{14}	0.08348
F_2	0	F_{15}	0
F_3	0	F_{16}	0.005287
F_4	0	F_{17}	5.479817
F_5	0.003549	F_{18}	0.002233
F_6	0	F_{19}	0.005198
F_7	0.000333	F_{20}	0.001165
F_8	0.000343	F_{21}	0.00192
F_9	0	F_{22}	24.07168
F_{10}	0	F_{23}	0
F_{11}	0	F_{24}	9.643301
F_{12}	0	F_{25}	4.836266
F_{13}	0.290247		

6 Results and comparisons

In addition to the above measures and results, the proposed RCSA algorithm is compared with recently developed state-of-the-art methods such as PSO (Kennedy and Eberhart 1995), IPOP-CMA-ES (Auger and Hansen 2005), CHC (Eshelman 1991; Eshelman and Schaffer 1993), SSGA (Fernandes and Rosa 2001; Mülenbein and Schlierkamp-Voosen 1993), SS-BLX (Herrera et al. 2006), SS-Arit (Laguna and Marti 2003), DE-Bin (Price et al. 2005) and SaDE (Qin and Suganthan 2005) as in Table 5. The comparisons indicate that the RCSA outperforms all other algorithms in terms of the average error except for the F_{24}.

Furthermore, the rank of the average error for different algorithms of each test function is reported, where the best value for the test function takes rank 1, worst value takes rank 10 and the other values are ranked between 1 and 10. As indicated from Table 5, we can say that the proposed RCSA algorithm surpasses all other algorithms on average.

Beside the use of the statistical measures for algorithm validations such as comparison of the proposed RCSA algorithm with other recent algorithms in terms of calculating the average error as well as calculations of best, mean and median results, we additionally apply saving of the fitness, $S_{fitness}$, in case of incorporating the RSS phase and without it. $S_{fitness}$ is calculated as follows:

Table 7 Wilcoxon test for comparison results in Table 5

Compared methods		Solution evaluations			
Algorithm 1	Algorithm 2	R^-	R^+	ρ-value	Best method
RCSA	PSO	276	0	0.000027	RCSA
RCSA	IPOP-CMA-ES	210	0	0.000089	RCSA
RCSA	CHC	300	0	0.000018	RCSA
RCSA	SSGA	231	0	0.000060	RCSA
RCSA	SS-BLX	325	0	0.000012	RCSA
RCSA	SS-Arit	325	0	0.000012	RCSA
RCSA	DE-Bin	210	0	0.000089	RCSA
RCSA	DE-Exp	190	0	0.000132	RCSA
RCSA	SaDE	190	0	0.000132	RCSA

$$S_{fitness} = \frac{F^O - F^{ORSS}}{F^O} \times 100 \tag{18}$$

where F^{ORSS}, F^O are the optimal objective value with and without RSS phase, respectively.

Table 6 demonstrates that there are a significant saving for the functions F_{17}, F_{22}, F_{24}, F_{25} and slight saving for the functions F_1, F_5, F_7, F_8, F_{13}, F_{14}, F_{16}, F_{18}–F_{21}. So, we conclude that the incorporating of the RSS phase improves the performance of the proposed RCSA algorithm through achieving a significant reduction in the optimal objective value as 44.5173% compared to the proposed approach without RSS phase.

In this subsection, a comparative study has been carried out to evaluate the performance of the proposed RCSA algorithm concerning the hybridization, closeness to optimal solution and computational time. On one hand, pure algorithms suffer from reaching an optimal solution in a reasonable time. Also, the sinking into premature convergence may be occurs in some of pure algorithms. Consequently, our hybridization algorithm has twofold features; avoiding the premature convergence and enclosing the optimum solution through using RSS phase by the means of the lower and upper approximations. On the other hand, the proposed RCSA algorithm is highly competitive when comparing it with the other methods in terms of the statistical measures. So the use of the hybrid approach has a great potential for solving global optimization problems.

6.1 Performance assessment

This section is devoted to assess the performance of the proposed algorithm using the Wilcoxon signed ranks test. The Wilcoxon signed ranks test is a nonparametric procedure used in a hypothesis testing situation involving a design with two samples (Joaquín et al. 2001). It is a pair-wise test that aims to detect significant differences between the behaviors of two methods. It is associated with ρ-value, where ρ is the probability of the null hypothesis being true. The result of the test is returned in $\rho < 0.05$ indicates a rejection of the null hypothesis, while $\rho > 0.05$ indicates a failure to reject the null hypothesis. The R^+ is the sum of positive ranks, while R^- is the sum of negative ranks. In Table 7, we present the results of the Wilcoxon signed-rank test for RCSA compared against PSO, IPOP-CMA-ES, CHC, SSGA, SS-BLX, SS-Arit, DE-Bin, DE-Exp and SaDE. We can conclude from Table 7 that the proposed RCSA is a significant algorithm and it is better than the other algorithms.

6.2 Large-scale test functions

To assess the performance of the proposed algorithm, we apply the proposed algorithm on a large-scale test functions for 1000 dimension. The large-scale test functions were proposed in the IEEE CEC 2010 (Tang et al. 2009) and also used in IEEE CEC 2012. Due to space limitation, the proposed algorithm is tested on five test functions of the IEEE CEC 2010.

In the other hand the proposed algorithm is compared with seven different algorithms, where the results of the seven comparative algorithms are taken from Gaoji et al. (2016). The selected compared algorithms are defined as follows: joint operations algorithm (JOA) (Gaoji et al. 2016), free search (FS) (Penev 2014), social- based algorithm (SBA) (Ramezani and Lotfi 2013), particle swarm optimizer with a diversity enhancing mechanism and neighborhood search strategies (DNSPSO) (Hui et al. 2013), dynamic multi-swarm particle swarm optimizer with a cooperative learning strategy (D-PSO-C) (Xu et al. 2015), dynamic group-based differential evolution (GDE) (Han et al. 2013) and sinusoidal differential evolution (SinDE) (Draa et al. 2015).

The results for large-scale test functions are reported in Table 8, where the best, worst, mean and standard deviation (Std.) are reported over 20 runs. From Table 8, it can be noted that RCSA outperforms the other algorithm in the view of statistical measures.

6.3 Engineering design problems

This section is devoted to validate the proposed RCSA for solving engineering design problems. Since these design problems involve different constraints, so the penalty function method is employed (Coello Coello 2002). By employing the penalty method, the constrained optimization problem can be converted to an unconstrained one and then the proposed RCSA algorithm can be implemented.

Table 8 Comparison among different algorithms on CEC 2010 functions

Fun.	Metric	FS	SBA	DNSPSO	D-PSO-C	GDE	SinDE	JOA	RCSA
f_1	Best	1.04E9	2.13E6	7.24E5	2.19E5	5.71E5	6.88E−7	1.54E−21	4.0389E−7
	Worst	1.58E9	2.80E6	9.37E6	1.02E6	1.86E7	3.16E−1	4.87E−17	3.1648E−5
	Mean	1.28eE9	2.43E6	3.18E6	6.12E5	5.07E6	2.18E−2	3.45E−19	1.0374E−5
	Std.	8.44E7	1.92E5	2.69E6	1.88E5	5.18E6	8.14E−2	1.26E−18	1.2513E−5
f_2	Best	9.02E3	7.92E3	6.01E3	1.26E3	6.45E3	1.20E3	7.51E2	4.7726E−4
	Worst	1.38E4	8.99E3	6.85E3	2.14E3	7.20E3	1.41E3	9.63E2	1.6704
	Mean	1.01E4	8.26E3	6.46E3	1.68E3	6.93E3	1.31E3	8.42E2	0.3580
	Std.	3.42E2	1.86E2	1.67E2	2.40E2	2.32E2	7.16E1	6.62E1	7.345E1
f_3	Best	2.06E1	1.90E1	1.83E1	1.14E1	1.93E1	2.25E0	2.16E0	7.5E−3
	Worst	2.11E1	1.96E1	1.95E1	1.59E1	2.02E1	2.76E0	2.72E0	3.42E−2
	Mean	2.09E1	1.95E1	1.93E1	1.33E1	1.96E1	2.48E0	2.47E0	1.85E−2
	Std.	1.94E−2	4.36E−2	1.30E−1	1.30E0	2.52E−1	3.43E−1	3.16E−1	1.19E−2
f_4	Best	1.48E12	2.08E11	9.76E11	3.72E12	7.60E11	1.18E12	1.41E11	8.3322E10
	Worst	2.69E12	6.10E11	8.99E12	1.96E13	1.82E12	3.25E12	3.23E11	5.6982E11
	Mean	2.04E12	3.71E11	2.46E12	8.16E12	1.34E12	1.71E12	2.45E11	3.1231E11
	Std.	2.85E11	1.12E11	2.02E12	5.19E12	3.32E11	6.29E11	7.58E10	1.4524E11
f_5	Best	9.15E7	2.74E8	1.49E8	5.21E7	7.36E7	4.08E7	3.88E7	1.2018E3
	Worst	1.46E8	4.03E8	3.99E8	3.51E8	2.02E8	6.67E7	6.87E7	6.3036E3
	Mean	1.11E8	3.32E8	2.63E8	2.95E8	1.26E8	5.44E7	4.85E7	4.5977E3
	Std.	1.42E7	4.43E7	5.82E7	9.70E7	4.03E7	7.43E6	1.17E7	2.9409E3

6.3.1 Himmelblau's design problem

The Himmelblau design problem was originally proposed by Himmelblau (1972) and it has been considered as a benchmark non-linear constrained optimization problem. On the other hand many authors have been tested another variation of this problem (named as version II) (Omran and Salman 2009), where a parameter 0.0006262 has been taken as 0.00026 (typeset bold in the constraint g_1). These problems can be formally defined as follows:

Version I:

$$\begin{aligned}
&\text{Min } F(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141\\
&\textit{subject to}:\\
&\qquad g_1(\mathbf{x}) = 85.334407 + 0.0056858x_2x_5 + \mathbf{0.0006262}x_1x_4 - 0.002205x_3x_5\\
&\qquad g_2(\mathbf{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2\\
&\qquad g_3(\mathbf{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.00190853x_3x_4\\
&\qquad 0 \le g_1(\mathbf{x}) \le 92, 90 \le g_2(\mathbf{x}) \le 110, 20 \le g_3(\mathbf{x}) \le 25,\\
&\qquad 78 \le x_1 \le 102, 33 \le x_2 \le 45, 27 \le x_i \le 45, \quad i = 3,4,5.
\end{aligned} \tag{19}$$

Version II:

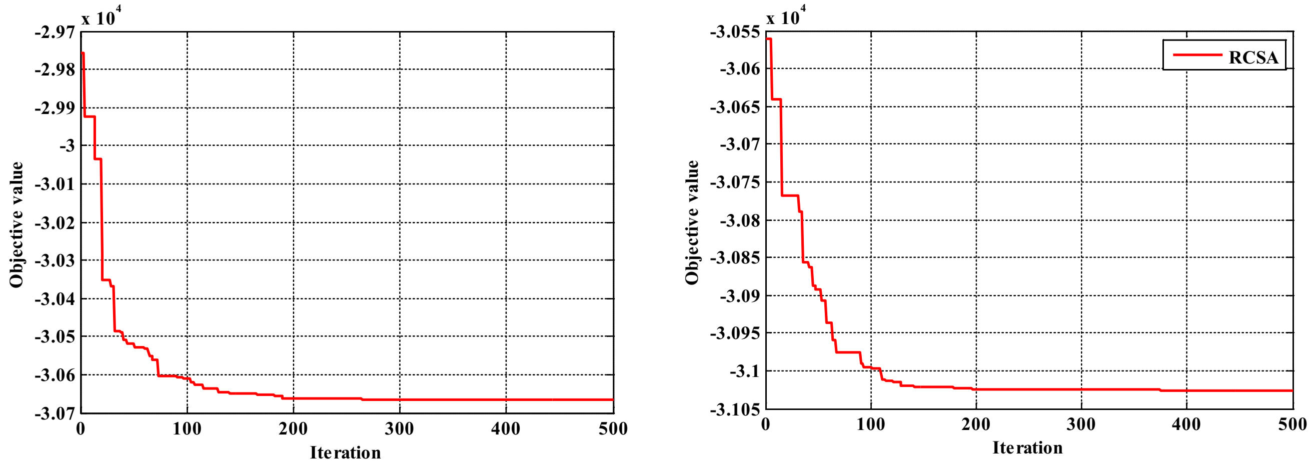
$$\begin{aligned}
&\text{Min } F(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141\\
&\textit{subject to}:\\
&\qquad g_1(\mathbf{x}) = 85.334407 + 0.0056858x_2x_5 + \mathbf{0.00026}x_1x_4 - 0.002205x_3x_5\\
&\qquad g_2(\mathbf{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2\\
&\qquad g_3(\mathbf{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.00190853x_3x_4\\
&\qquad 0 \le g_1(\mathbf{x}) \le 92, 90 \le g_2(\mathbf{x}) \le 110, 20 \le g_3(\mathbf{x}) \le 25,\\
&\qquad 78 \le x_1 \le 102, 33 \le x_2 \le 45, 27 \le x_i \le 45, \quad i = 3,4,5.
\end{aligned} \tag{20}$$

The proposed RCSA algorithm has been tested on both the versions of this problem through finding the best, median, mean and worst values. Further the proposed RCSA algorithm is compared with prominent different algorithms that reported in Deb (2000), He et al. (2004), Lee and Geem (2005), Dimopoulos (2007), Gandomi et al. (2013) and Mehta and Dasgupta (2012) for the first version and Omran and Salman (2009), Coello (2000), Fesanghary et al. (2008) and Hu et al. (2003) for the second version as in Table 9. Table 9 presents the statistical results for the two versions in terms of finding the values of best, median,

Table 9 Statistical results for the Himmelblau's problem

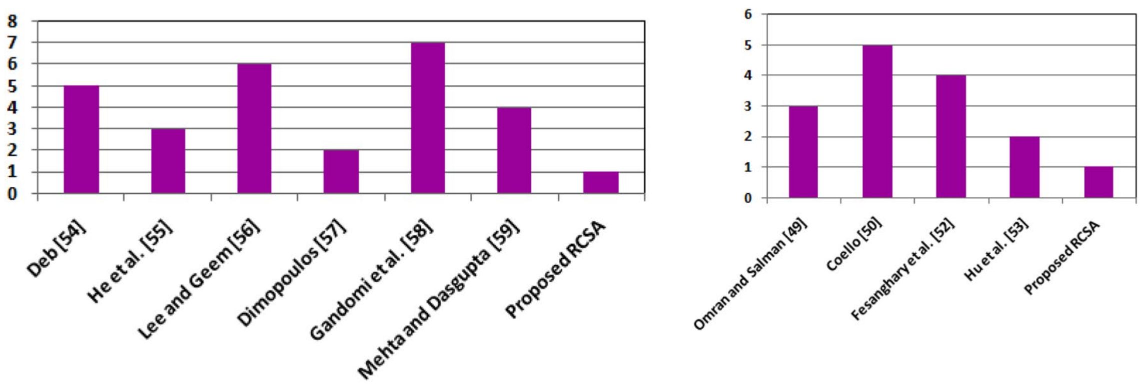
Version	Methods	Best	Median	Mean	Worst	Std.
I	Deb (2000)	− 30,665.537	− 30,665.535	NA	− 29,846.654	NA
	He et al. (2004)	− 30,665.539	NA	− 30,643.989	NA	70.043
	Lee and Geem (2005)	− 30,665.500	NA	NA	NA	NA
	Dimopoulos (2007)	− 30,665.54	NA	NA	NA	NA
	Gandomi et al. (2013)	− 30,665.2327	NA	NA	NA	11.6231
	Mehta and Dasgupta (2012)	− 30,665.538741	NA	NA	NA	NA
	Proposed RCSA	− 30,665.545314	− 30,665.545314	− 30,665.544377	− 30,665.542970	0.001283
II	Omran and Salman (2009)	− 31,025.55626	NA	− 31,025.556264	NA	NA
	Coello (2000)	− 31,020.859	− 31,017.21369	− 30,984.240703	− 30,792.407737	73.633536
	Fesanghary et al. (2008)	− 31,024.3166	NA	NA	NA	NA
	Hu et al. (2003)	− 31,025.56142	NA	− 31,025.561420	NA	0
	Proposed RCSA	− 31,025.568575	− 31,025.568512	− 31,025.559812	− 31,025.507752	0.018780

NA not available



(a) version I **(b)** version II

Fig. 6 Convergence curves for the Himmelblau's design problem



(a) version I **(b)** version II

Fig. 7 Ranking of the best solutions for the Himmelblau's design problem

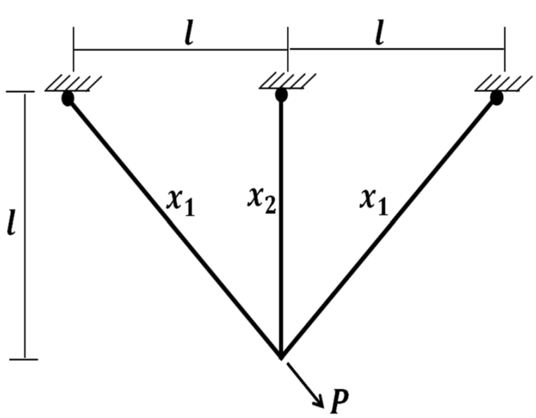


Fig. 8 Architecture of three-bar truss design problem

mean, worst and the standard deviation (Std.) obtained by 30 runs, where the optimal solution for the version I is **x** = [78.000122 33.000129 29.995023 44.999358 36.776087] and the objective function value is – 30,665.545314. On the other hand the optimal solution for the version II is **x** = [78.000094 33.000075 27.070838 44.999957 44.969327]with the objective function value is – 31,025.568575. Based on the depicted comparisons in Table 9, we can see that the proposed RCSA algorithm outperforms the other methods, where it gives better solutions for the two versions than the other algorithms.

Figure 6 illustrates the convergence curves for the best objective value obtained by the proposed RCSA algorithm for the two versions of the Himmelblau design problem. It can be seen the convergence curve rapidly convergent to

Table 10 Comparison between the proposed RCSA and different algorithms for three-bar truss design problem

Algorithm	Best	Mean	Median	Worst	SD
Proposed RCSA	**263.895843376**	**263.895843377**	**263.895843378**	**263.895843378**	8.0468107E−010
Hui et al. (2010)	263.89584338	263.89584338	NA	263.89584338	**4.5E−10**
Ray and Liew (2003)	263.89584654	263.90335672	NA	263.96975638	1.3E−02

the optimal solution in less than 300 iterations for version I and less than 200 iterations for version II.

Figure 7 provides the ranks for the different algorithms, where the best value takes the order one; the second best takes the order two; and so on. As shown from Fig. 7, we can see that the proposed RCSA algorithm gives the better rank over all other algorithms.

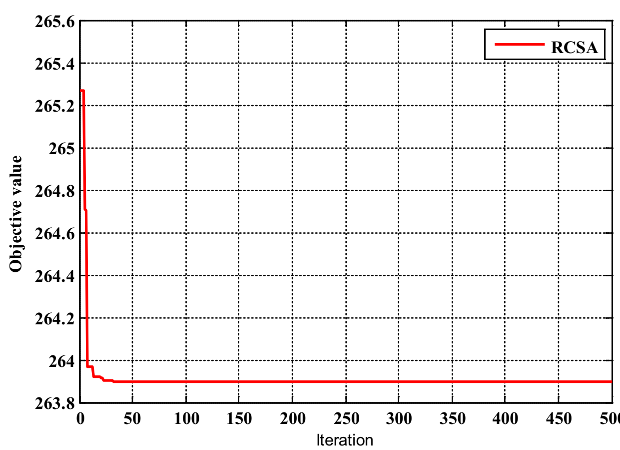


Fig. 9 Convergence behavior of the RCSA of three-bar truss design problem

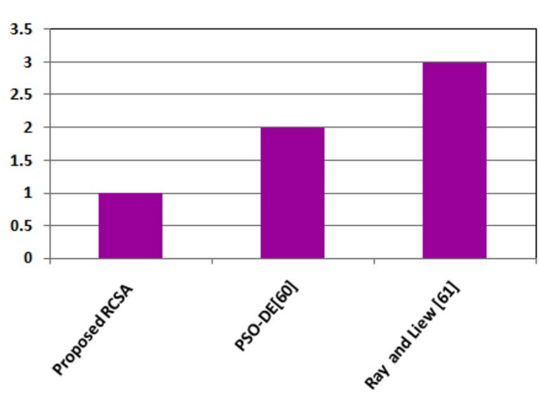


Fig. 10 Ranking of the optimum solutions for the three-bar truss design problem

6.3.2 Three-bar truss design problem

The three-bar truss design problem is to minimize the volume of a statistically loaded three-bar truss as the objective function and subject to stress (σ) constraints on each of the truss members by adjusting cross sectional areas (x_1 and x_2). The schematic of three-bar truss design problem is depicted in Fig. 8. This optimization problem is defined as follows:

$$\text{Min}\, F(\mathbf{x}) = (2\sqrt{2}x_1 + x_2) \times l$$

subject to:

$$g_1(\mathbf{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \le 0$$
$$g_2(\mathbf{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \le 0$$
$$g_3(\mathbf{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \le 0$$
$$0 \le x_1, x_2 \le 1, l = 100\,\text{cm}, P = 2\,\text{kN/cm}^2, \sigma = 2\,\text{kN/cm}^2. \tag{21}$$

The results of the proposed algorithm are obtained for three-bar truss design problem. The proposed RCSA yields the optimal solution and constraints value as follow: $\mathbf{x} = [0.7886751333, 0.4082482940]$ and $g(\mathbf{x}) = [0, -1.4641016110, -0.5358983889]$. In addition, the comparisons between the proposed RCSA algorithm and other different algorithms (i.e., PSO-DE; Hui et al.

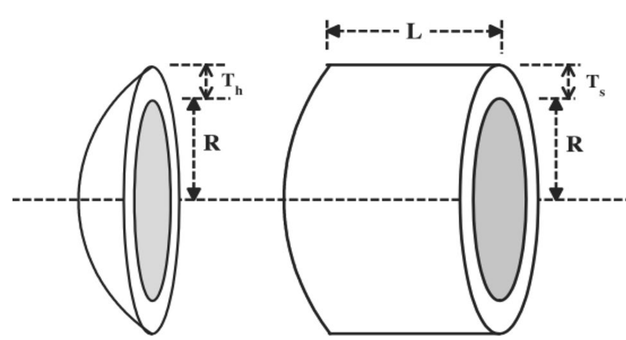


Fig. 11 Architecture of pressure vessel design problem

Table 11 Statistical results of different methods for pressure vessel

Method	Best	Mean	Worst	SD	Median
Sandgren (1988)	8129.1036	N/A	N/A	N/A	NA
Kannan and Kramer (1994)	7198.0428	N/A	N/A	N/A	NA
Deb and Gene (1997)	6410.3811	N/A	N/A	N/A	NA
Coello (2000)	6288.7445	6293.8432	6308.1497	7.4133	NA
Coello and Montes (2002)	6059.9463	6177.2533	6469.3220	130.9297	NA
He and Wang (2007)	6061.0777	6147.1332	6363.8041	86.4545	NA
Montes and Coello (2008)	6059.7456	6850.0049	7332.8798	426.0000	NA
Kaveh and Talatahari (2010)	6059.7258	6081.7812	6150.1289	67.2418	NA
Kaveh and Talatahari (2009)	6059.0925	6075.2567	6135.3336	41.6825	NA
Gandomi et al. (2013)	6059.714	6447.7360	6495.3470	502.693	NA
Cagnina et al. (2008)	6059.714335	6092.0498	NA	12.1725	NA
Coello Coello et al. (2010)	6059.7208	6440.3786	7544.4925	448.4711	6257.5943
He et al. (2004)	6059.7143	6289.92881	NA	305.78	NA
Akay and Karaboga (2012)	6059.714339	6245.308144	NA	205	NA
Garg (2014)	5885.403282	5887.557024	5895.126804	2.745290	5886.14928
Proposed RCSA	6059.606944	6059.844857	6061.034418	0.0582763	6059.606944

NA not available

Table 12 The optimal design variables with their objective values

Method	x_1	x_2	x_3	x_4	$F(\mathbf{x})$
Sandgren (1988)	1.125000	0.625000	47.700000	117.701000	8129.1036
Kannan and Kramer (1994)	1.125000	0.625000	58.291000	43.690000	7198.0428
Deb and Gene (1997)	0.937500	0.500000	48.329000	112.67900	6410.3811
Coello (2000)	0.812500	0.437500	40.323900	200.000000	6288.7445
Coello and Montes (2002)	0.812500	0.437500	42.097398	176.654050	6059.946
He and Wang 2007	0.812500	0.437500	42.091266	176.746500	6061.0777
Montes and Coello (2008)	0.812500	0.437500	42.098087	176.640518	6059.7456
Kaveh and Talatahari (2010)	0.812500	0.437500	42.103566	176.573220	6059.0925
Kaveh and Talatahari (2009)	0.812500	0.437500	42.098353	176.637751	6059.7258
Zhang and Wang (1993)	1.125000	0.625000	58.290000	43.6930000	7197.7000
Cagnina et al. (2008)	0.812500	0.437500	42.098445	176.6365950	6059.714335
Coello Coello et al. (2010)	0.812500	0.437500	42.098400	176.6372000	6059.7208
He et al. (2004)	0.812500	0.437500	42.098445	176.6365950	6059.7143
Lee and Geem (2005)	1.125000	0.625000	58.278900	43.75490000	7198.433
Montes et al. (2007)	0.812500	0.437500	42.098446	176.6360470	6059.701660
Hu et al. (2003)	0.812500	0.437500	42.098450	176.6366000	6059.7151717976
Gandomi et al. (2013)	0.812500	0.437500	42.0984456	176.6365958	6059.7143348
Akay and Karaboga (2012)	0.812500	0.437500	42.098446	176.636596	6059.714339
Garg (2014)	0.7781977	0.3846656	40.3210545	199.9802367	5885.4032828
Proposed RCSA	0.812500	0.437500	42.100204	176.614800	6059.606944

NA not available

2010; Ray and Liew 2003) are presented in Table 10. Table 10 demonstrates that the proposed algorithm outperform the other algorithms in terms of quality of the obtained solution, where the better solution among all algorithm is highlighted in boldface. Therefore, it can be concluded that the proposed RCSA algorithm is a good alternative algorithm for design problems. Further, the convergence behavior of the RCSA for obtaining the best

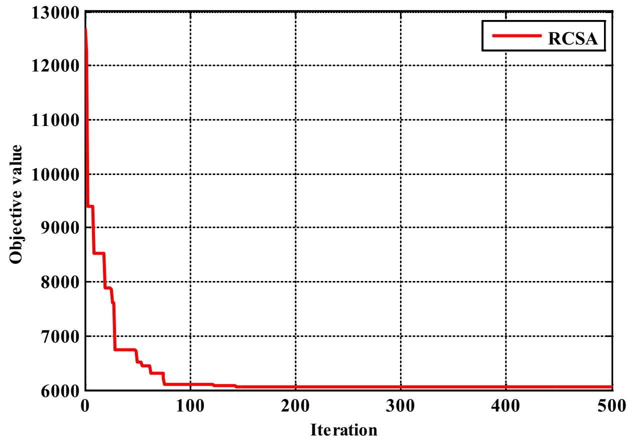


Fig. 12 Convergence behavior of the pressure vessel design problem

objective value of the three-bar truss design problem is illustrated in Fig. 9. We can see that the proposed algorithm converges rapidly to the optimal solution in less than 40 iterations.

Beside these results, it can see that the proposed RCSA provides the better result than all other algorithms and then its result comes in the first rank, while PSO-DE provides the second best solution as the second rank result, then Ray and Liew comes in third rank. In general, the results of these algorithms are depicted according these ranks in Fig. 10. Figure 10 illustrates that the proposed RCSA algorithm finds the better rank over all other algorithms.

6.3.3 Pressure vessel design problem

The goal of the pressure vessel design is to minimize the total cost (i.e., the cost of material, forming and welding) (Sandgren 1988) of a cylindrical vessel that is capped at both ends by hemi-spherical heads as shown in Fig. 11. Using rolled steel plate, the shell is made in two halves that are joined by two longitudinal welds to form a cylinder. There are four design variable associated with it, namely the thickness of the pressure vessel, $T_s = x_1$, thickness of the head, $T_h = x_2$, inner radius of the vessel, $R = x_3$, and length of the vessel without heads, $L = x_4$, i.e., the variable vectors are

Fig. 13 Ranking of the optimum solutions for the pressure vessel design problem

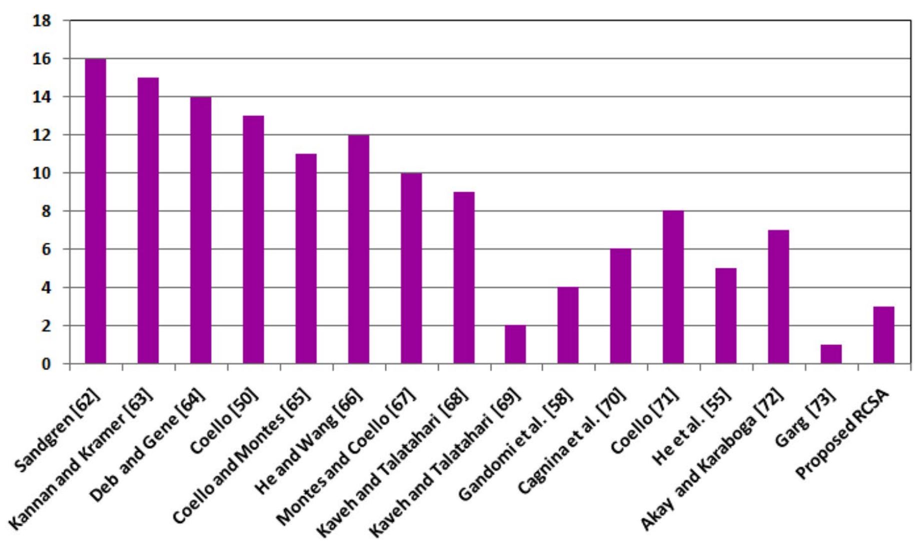


Fig. 14 Schematic of the speed reducer design problem

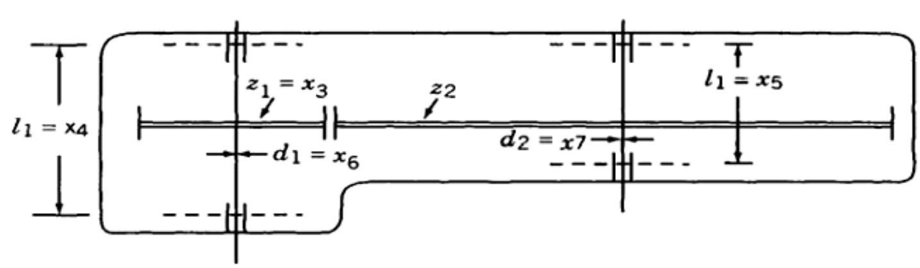


Table 13 Comparing of the speed reducer design problem results of RCSAwith other algorithms

Algorithm	Best	Mean	Median	Worst	SD
Proposed RCSA	2994.381855	2994.381855	2994.381855	2994.381855	0.0000
Hui et al. (2010)	2996.348167	2996.348174	NA	2996.348204	6.4E−06
Ray and Liew (2003)	2994.744241	3001.758264	NA	3009.964736	4.0E+00
Rao and Xiong (2005)	3000.959715	NA	NA	NA	NA
Cagnina et al. (2008)	2996.347849	NA	NA	NA	NA
Tosserams et al. (2007)	2996.645783	NA	NA	NA	NA
Lu and Kim (2010)	3019.583365	NA	NA	NA	NA

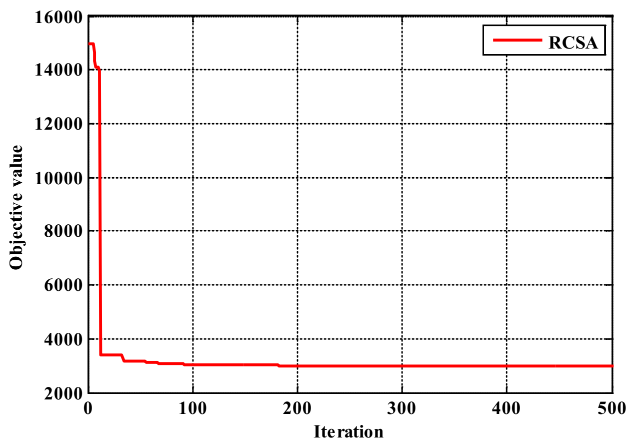


Fig. 15 Convergence behavior of the speed reducer design problem

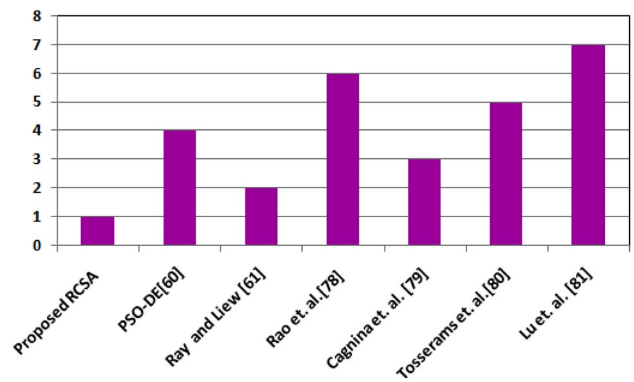


Fig. 16 Ranking of the optimum solutions for the speed reducer design problem

given (in inches) by $\mathbf{x} = (T_s, T_h, R, L_s) = (x_1, x_2, x_3, x_4)$. Then, the problem is formulated mathematically as follows:

$$\text{Min } F(\mathbf{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

subjec to:

$$\begin{aligned}
&g_1(\mathbf{x}) = -x_1 + 0.0193x_3 \le 0,\\
&g_2(\mathbf{x}) = -x_3 + 0.00954x_3 \le 0,\\
&g_3(\mathbf{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1,296,000 \le 0,\\
&g_4(\mathbf{x}) = x_4 - 240 \le 0,\\
&1 \times 0.0625 \le x_1, x_2 \le 99 \times 0.0625,\ 10 \le x_3, x_4 \le 99 \times 200.
\end{aligned} \tag{22}$$

The optimal solution obtained by implementing the proposed RCSA algorithm is $\mathbf{x} = (0.8125\ 0.437500\ 42.100204\ 176.614800)$ with corresponding function value equal to $f(\mathbf{x}) = 6059.606944$ and in addition the constraints are calculated (i.e., $[g_1\ g_2\ g_3\ g_4] = [3.394885\text{E}-5\ \ -0.037548\ \ -0.000278\ \ -63.385199]$). Table 11 outlines the statistical results of the proposed RCSA algorithm in terms of obtaining the best, mean, worst, std. and median over 30 run. Moreover these results are compared with other algorithms (Sandgren 1988; Kannan and Kramer 1994; Deb and Gene 1997; Coello and Montes 2002; He and Wang 2007; Montes and Coello 2008; Kaveh and Talatahari 2010, 2009; Cagnina et al. 2008; Coelho 2010; Akay and Karaboga 2012; Garg 2014; Zhang and Wang 1993; Montes et al. 2007). Table 11 shows that the proposed RCSA algorithm outperforms the other optimization algorithms. Although the produced solution by Garg (2014) is better than the proposed RCSA algorithm, the solution of design variables are violated with the design variables restrictions (i.e., x_1 is discrete).

Table 12 provides the comparisons between the proposed RCSA algorithm and other algorithms in terms of finding the optimal design variables with their corresponding function value. We can see that the produced solution of the design variables by Garg (2014) are violated with the design variables restrictions. However, the proposed RCSA algorithm provides better objective function than those provided by the literature. Table 12 indicates that the proposed RCSA algorithm is more robust than the other methods.

The convergence curve of the proposed RCSA algorithm for the pressure vessel design problem is provided Fig. 12. The convergence behavior indicates that the proposed

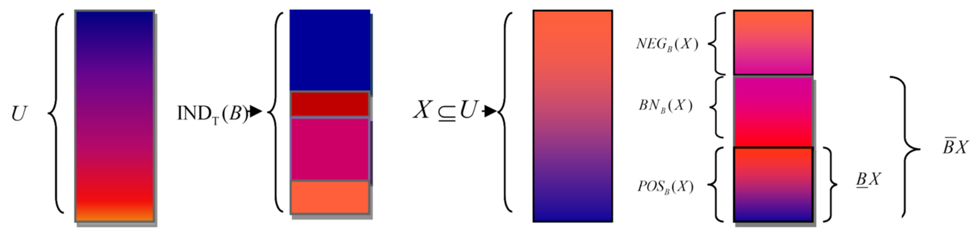


Fig. 17 Definitions regarding rough set approximations

RCSA algorithm finds the optimal solution in less than 150 iterations.

From Fig. 13, we see that the proposed RCSA algorithm takes the order three, but Garg (2014) and Kaveh and Talatahari (2009) take the order one and two respectively. On the hand, the Garg (2014) violates the bound restriction for the variable x_1 and Kaveh and Talatahari (2009) provides the value of constraint as $g_1 = 9.9\text{E}{-5}$ which is greater than that provides by the proposed RCSA algorithm. Consequently, the proposed RCSA algorithm is still better for this problem.

6.3.4 Speed reducer design problem

The main goal of the speed reducer design problem is to minimize the total weight of the speed reducer while satisfying some constraints. This design problem has a rather difficult to detect feasible space as reported in Golinski (1973), where the constraints include limitations on the bending stress of gear teeth, surface stress, transverse deflections of shafts 1 and 2 due to transmitted force, and stresses in shafts 1 and 2. Figure 14 shows the schematic shape of the speed reducer design problem in which seven unknown of design variables is showed, where the design of the speed reducer is considered by the face width, $b = x_1$, module of teeth, $m = x_2$, number of teeth on pinion, $z = x_3$, length of shaft 1 between bearings, $l_1 = x_4$, length of shaft 2 between bearings, $l_2 = x_5$, diameter of shaft 1, $d_1 = x_6$, and diameter of shaft 2 $d_2 = x_7$ i.e., the variable vectors are given by $\mathbf{x} = (b, m, z, l_1, l_2, d_1, d_2) = (x_1, x_2, x_3, x_4, x_5, x_6, x_7)$. Then, the problem is formulated mathematically as follows:

$$
\begin{aligned}
\text{Min } F(\mathbf{x}) = {} & 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\
& - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) \\
& + 0.7854(x_4x_6^2 + x_5x_7^2) \\
& \text{subjec to:} \\
& g_1(\mathbf{x}) = \frac{27}{x_1x_2^2x_3} - 1 \le 0, \\
& g_2(\mathbf{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0, \\
& g_3(\mathbf{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0, \\
& g_4(\mathbf{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0, \\
& g_5(\mathbf{x}) = \frac{\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9(10^6)}}{110x_6^3} - 1 \le 0, \\
& g_6(\mathbf{x}) = \frac{\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5(10^6)}}{85x_7^3} - 1 \le 0, \\
& g_7(\mathbf{x}) = \frac{x_2x_3}{40} - 1 \le 0, \\
& g_8(\mathbf{x}) = \frac{5x_2}{x_1} - 1 \le 0, \\
& g_9(\mathbf{x}) = \frac{x_1}{12x_2} - 1 \le 0, \\
& g_{10}(\mathbf{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0, \\
& g_{10}(\mathbf{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0, \\
& 2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, \\
& 7.3 \le x_4, x_5 \le 8.3, 2.9 \le x_6 \le 3.9, 5 \le x_7 \le 5.5.
\end{aligned} \tag{23}
$$

The best objective value obtained by proposed RCSA algorithm is $F(\mathbf{x}) = 2994.381855$ and the optimal solution of the design variables are $\mathbf{x} = (3.500006\ 0.700001\ 17.000000\ 7.300562\ 7.715339\ 3.350260\ 5.286657)$. Table 13 provides the best solution obtained by RCSA for speed reducer design problem over 30 independent runs, where the comparison of the statistical results obtained by RCSA and other algorithms is are shown. As we can see, the results show that the proposed RCSA algorithm produces promising results in comparison with the other methods of the speed reducer design problem in terms of obtaining the best, mean, median and the worst. In addition the minimal value of standard deviation (Std.) denotes the high robustness of RCSA. From Fig. 15 demonstrates the convergence behavior of the proposed RCSA algorithm where it finds the optimal solution in less than 200 iterations. On the other hand Fig. 16 provides the comparisons of ranks for different algorithms, where the proposed algorithm gives the better rank thus it outperforms the other algorithms.

7 Conclusions

We concluded that the integrated RCSA has improved the quality of the found solutions and also guaranteed the faster converge to the optimal solution. In the RCSA, CSA phase is presented in the first stage to provide the initial optimal solution of the optimization problem while the RSS phase is introduced as a second stage to enhance the exploitation search. However the flight length of the traditional CSA is fixed, it may produce unsatisfactory solution. So a dynamic flight length behavior is introduced with the aim of eliciting values from the interval $[fl_{\min}, fl_{\max}]$ to enhance the exploration process. The proposed RCSA algorithm is investigated on 30 benchmark problems of IEEE CEC 2005, IEEE CEC 2010 and 4 engineering design problems. The obtained results by RCSA are compared with different algorithms from the literature. The simulations showed that the incorporation of RSS provides an important modification on the CSA. In comparison with the classical CSA and other algorithms from the literature, it seems that the RCSA performed significantly well. The superior results of RCSA on the benchmark problems and engineering design problems showed its applicability for complex real-world problems. The main reason of the superior performance of RCSA lies behind the RSS which helps in breaking new promising regions by the means of the lower and upper approximations and thus it can refine the convergence rate of the algorithm and avoid the sucking in the local optima. Therefore, we can conclude that the proposed RCSA can handle engineering optimization problems efficiently and effectively.

The future work will be focused on applying the methodology of RCSA to solve the multi-objective problems, mixed-type problems, and discrete optimization problems in smart and complex applications (Abdelaziz et al. 2018; Darwish et al. 2017; Elhoseny et al. 2018b, c; Sajjad et al. 2017; Shehab et al. 2018).

Appendix 1: Rough set theory definitions

Definition A.1 (*Information system*) An information system (IS) is denoted as a triplet $T = (U, A, f)$, where U is a non-empty finite set of objects and A is a non-empty finite set of attributes. An information function f maps an object to its attribute, i.e., $f_a : U \rightarrow V_a$ for every $a \in A$, where V_a is the value set of attribute a. A posteriori knowledge (denoted by d) is expressed by one distinguished attribute. A decision system is an IS with the form $DT = (U, A \cup \{d\}, f)$, where $d \notin A$ is used as supervised learning. The elements of A are called conditional attributes.

Definition A.2 (*Indiscernibility*) For an attribute set $B \subseteq A$, the equivalence relation induced by B is called a B-indiscernibility relation, i.e., $\mathrm{IND_T}(B) = \left\{(x, y) \in U^2 \mid \forall a \in B, f_a(x) = f_a(y)\right\}$. The equivalence classes of the B-indiscernibility relation are denoted as $I_B(x)$.

Definition A.3 (*Set approximation*) Let $X \subseteq U$ and $B \subseteq A$ in an IS, the B-lower approximation of X is the set of objects that belongs to X with certainty, i.e., $\underline{B}X = \left\{x \in U \mid I_B(x) \subseteq X\right\}$. The B-upper is the set of objects that possibly belongs to X, where $\bar{B}X = \left\{x \in U \mid I_B(x) \cap X \neq \phi\right\}$.

Definition A.4 (*Reducts*) If $X^1_{DT}, X^2_{DT}, \ldots, X^r_{DT}$ are the decision classes of DT, the set $POS_B(d) = \underline{B}X^1 \cup \underline{B}X^2 \cup \cdots \cup \underline{B}X^r$ is the B-positive region of DT. A subset $B \subseteq A$ is a set of relative reducts of DT if and only if $POS_B(d) = POS_C(d)$ and $POS_{B-\{b\}}(d) \neq POS_C(d)$, $\forall b \in B$. In the same way, $POS_B(X)$, $BN_B(X)$ and $NEG_B(X)$ are defined below (see Fig. 17).

- $POS_B(X) = \underline{B}X \Rightarrow$ certainly member of X
- $NEG_B(X) = U - \bar{B}X \Rightarrow$ certainly non member of X
- $BN_B(X) = \bar{B}X - \underline{B}X \Rightarrow$ possibly member of X.

References

Abdelaziz A, Elhoseny M, Salama AS, Riad AM (2018) A machine learning model for improving healthcare services on cloud computing environment. Measurement 119:117–128

Akay B, Karaboga D (2012) Artificial bee colony algorithm for large-scale problems and engineering design optimization. J Intell Manuf 23(4):1001–1014

Alireza A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput Struct 169:1–12

Auger A, Hansen N (2005) A restart CMA evolution strategy with increasing population size. In: Proceedings of the 2005 IEEE congress on evolutionary computation, pp 1769–1776

Bartholomew-Biggs M (2008) Nonlinear optimization with engineering applications. Springer Optim Appl 19:1–14

Cagnina LC, Esquivel SC, Coello Coello CA (2008) Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica 32(3):319–326

Chijun Z, Yongjian Y, Zhanwei D, Chuang M (2016) Particle swarm optimization algorithm based on ontology model to support cloud computing applications. J Ambient Intell Humaniz Comput 7(5):633–638

Coelho LS (2010) Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. Expert Syst Appl 37(2):1676–1683

Coello CAC (2000) Use of a self -adaptive penalty approach for engineering optimization problems. Comput Ind 41(2):113–127

Coello Coello CA (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng 191(11):1245–1287

Coello Coello CA, Dhaenens C, Jourdan L (2010) Multi-objective combinatorial optimization: problematic and context. In: Coello Coello CA, Dhaenens C, Jourdan L (eds) Advances in multi-objective nature inspired computing. Studies in computational intelligence, vol 272. Springer, Berlin, Heidelberg, pp 1–21

Coello CAC, Montes EM (2002) Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inf 16(3):193–203

Darwish A, Hassanien AE, Elhoseny M, Sangaiah AK, Muhammad K (2017) The impact of the hybrid platform of internet of things and cloud computing on healthcare systems: opportunities, challenges, and open problems. J Ambient Intell Humaniz Comput. https://doi.org/10.1007/s12652-017-0659-1

Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Eng 186:311–338

Deb K, Gene AS (1997) A robust optimal design technique for mechanical component design. In: Dasgupta D, Michalewicz Z (eds) Evolutionary algorithms in engineering applications. Springer, Berlin, pp 497–514

Dimopoulos GG (2007) Mixed-variable engineering optimization based on evolutionary and social metaphors. Comput Methods Appl Mech Eng 196(4–6):803–817

Draa A, Bouzoubia S, Boukhalfa I (2015) A sinusoidal differential evolution algorithm for numerical optimisation. Appl Soft Comput 27:99–126

Elhoseny M, Tharwat A, Hassanien AE (2018a) Bezier curve based path planning in a dynamic field using modified genetic algorithm. J Comput Sci 25:339–350

Elhoseny M, Abdelaziz A, Salama AS, Riad AM, Muhammad K, Sangaiah AK (2018b) A hybrid model of Internet of Things and cloud computing to manage big data in health services applications. Future Gener Comput Syst 86:1383–1394

Elhoseny M, Ramírez-González G, Abu-Elnasr OM, Shawkat SA, Arunkumar N, Farouk A (2018c) Secure medical data transmission model for IoT-based healthcare systems. IEEE Access. https://doi.org/10.1109/ACCESS.2018.2817615

Eshelman LJ (1991) The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination. In: Rawlins GJE (ed) Foundations of genetic algorithms. Morgan Kaufmann, San Mateo, pp 265–283

Eshelman LJ, Schaffer JD (1993) Real-coded genetic algorithms and interval schemata. In: Whitley D (ed) Foundations of genetic algorithms. Morgan Kaufmann, San Mateo, pp 187–202

Fernandes C, Rosa A (2001) A study of non-random matching and varying population size in genetic algorithm using a royal road function. In: Proceedings of the 2001 congress on evolutionary computation, pp 60–66

Fesanghary M, Mahdavi M, Minary-Jolandan M, Alizadeh Y (2008) Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems. Comput Methods Appl Mech Eng 197(33–40):3080–3091

Gandomi A, Yang XS, Alavi A (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17–35

Gaoji S, Ruiqing Z, Yanfei L (2016) Joint operations algorithm for large-scale global optimization. Appl Soft Comput 38:1025–1039

Garg H (2014) Solving structural engineering design optimization problems using an artificial bee colony algorithm. J Ind Manag Optim 10(3):777–794

Golinski J (1973) An adaptive optimization system applied to machine synthesis. Mech Mach Theory 8(4):419–436

Han MF, Liao SH, Chang JY, Lin CT (2013) Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems. Appl Intell 39(1):41–56

He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 20(1):89–99

He S, Prempain E, Wu QH (2004) An improved particle swarm optimizer for mechanical design optimization problems. Eng Optim 36(5):585–605

Herrera F, Lozano M, Molina D (2006) Continuous scatter search: an analysis of the integration of some combination methods and improvement strategies. Eur J Oper Res 169(2):450–476

Himmelblau DM (1972) Applied nonlinear programming. McGraw-Hill, New York

Hu XH, Eberhart RC, Shi YH (2003) Engineering optimization with particle swarm. In: Proceedings of the 2003 IEEE swarm intelligence symposium, pp 53–57

Hui L, Zixing C, Yong W (2010) Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl Soft Comput 10(2):629–640

Hui W, Hui S, Changhe L, Shahryar R, Jeng-shyang P (2013) Swarm optimization with neighborhood search. Inf Sci 223:119–135

Jie H, Tianrui L, Chuan L, Hamido F, Yan Y (2017) Incremental fuzzy cluster ensemble learning based on rough set theory. Knowl Based Syst 132(15):144–155

Joaquín D, Salvador G, Daniel M, Francisco H (2001) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18

Kannan BK, Kramer SN (1994) An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. J Mech Des 116(2):318–320

Kaveh A, Talatahari S (2009) Engineering optimization with hybrid particle swarm and ant colony optimization. Asian J Civ Eng (Build Hous) 10(6):611–628

Kaveh A, Talatahari S (2010) An improved ant colony optimization for constrained engineering design problems. Eng Comput 27(1):155–182

Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IV IEEE international conference on neural networks, pp 1942–1948

Laguna M, Marti R (2003) Scatter search: methodology and implementation in C. Kluwer Academic Publishers, Dordrecht

Lee KS, Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Methods Appl Mech Eng 194(36–38):3902–3933

Li Y, Liao X, Zhao W (2009) A rough set approach to knowledge discovery in analyzing competitive advantages of firms. Ann Oper Res 168(1):205–223

Lu S, Kim HM (2010) A regularized inexact penalty decomposition algorithm for multidisciplinary design optimization problems with complementarity constraints. J Mech Des 132(4):1–12

Mehta VK, Dasgupta B (2012) A constrained optimization algorithm based on the simplex search method. Eng Optim 44(5):537–550

Metawa N, Hassana MK, Elhoseny M (2017) Genetic algorithm based model for optimizing bank lending decisions. Expert Syst Appl 80:75–82

Mohit J, Asha R, Vijander S (2017) An improved Crow Search Algorithm for high-dimensional problems. J Intell Fuzzy Syst 33:3597–3614

Montes EM, Coello CAC (2008) An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int J Gen Syst 37(4):443–473

Montes EM, Coello CAC, Reyes JV, Davila LM (2007) Multiple trial vectors in differential evolution for engineering design. Eng Optim 39(5):567–589

Mousa AA, Abd El-Wahed WF, RizkAllah RM (2011) A hybrid ant colony optimization approach based local search scheme for multiobjective design optimizations. Electr Power Syst Res 81:1014–1023

Mülenbein H, Schlierkamp-Voosen D (1993) Predictive models for the breeding genetic algorithm in continuous parameter optimization. Evol Comput 1(1):25–49

Omran MGH, Salman A (2009) Constrained optimization using CODEQ. Chaos Solitons Fractals 42(2):662–668

Pan QK, Sang HY, Duan JH, Gao L (2014) An improved fruit fly optimization algorithm for continuous function optimization problems. Knowl Based Syst 62:69–83

Pawlak Z (1982) Rough sets. Int J Comput Inform Sci 11:341–356

Penev K (2014) Free search-comparative analysis 100. Int J Metaheuristics 3(2):118–132

Price KV, Rainer M, Lampinen JA (2005) Differential evolution: a practical approach to global optimization. Springer, Berlin

Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of the 2005 IEEE congress on evolutionary computation, vol 2, pp 1785–1791

Ramezani F, Lotfi S (2013) Social-based algorithm (SBA). Appl Soft Comput 13:2837–2856

Rao SS (2009) Engineering optimization-theory and practice. Wiley, New York

Rao SS, Xiong Y (2005) A hybrid genetic algorithm for mixed discrete design optimization. J Mech Des 127(6):1100–1112

Ray T, Liew KM (2003) Society and civilization: an optimization algorithm based on the simulation of social behavior. IEEE Trans Evol Comput 7(4):386–396

Rizk-Allah RM (2016) Fault diagnosis of the high-voltage circuit breaker based on granular reduction approach. Eur J Sci Res 138(1):29–37

Rizk-Allah RM (2018) Hybridizing sine cosine algorithm with multi-orthogonal search strategy for engineering design problems. J Comput Des Eng 5:249–273

Rizk-Allah RM, Zaki EM, El-Sawy AA (2013) Hybridizing ant colony optimization with firefly algorithm for unconstrained optimization problems. Appl Math Comput 224:473–483

Rizk-Allah RM, Abdel-Mageed HM, El-Sehiemy RA, Abdel-Aleem SH, El-Shahat A (2017a) A new sine cosine optimization algorithm for solving combined non-convex economic and emission power dispatch problems. Int J Energy Convers 5(6):180–192

Rizk-Allah RM, El-Sehiemy RA, Deb S, Wang GG (2017b) A novel fruit fly framework for multi-objective shape design of tubular linear synchronous motor. J Supercomput 73(3):1235–1256

Rizk-Allah RM, El-Sehiemy RA, Wang GG (2018a) A novel parallel hurricane optimization algorithm for secure emission/economic load dispatch solution. Appl Soft Comput 63:206–222

Rizk-Allah RM, Hassanien AE, Bhattacharyya S (2018b) Chaotic crow search algorithm for fractional optimization problems. Appl Soft Comput. https://doi.org/10.1016/j.asoc.2018.03.019

Rubén AR, Manuel VR, Rodríguez-Ortiz JJ (2015) Genetic algorithms and Darwinian approaches in financial applications: a survey. Expert Syst Appl 42(21):7684–7697

Sajjad M, Nasir M, Muhammad K, Khan S, Jan Z, Sangaiah AK et al (2017) Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities. Future Gener Comput Syst. https://doi.org/10.1016/j.future.2017.11.013

Sandgren E (1988) Nonlinear integer and discrete programming in mechanical design. In: Proceedings of the ASME design technology conference, F.L. Kissimine, pp 95–105

Sayed GI, Hassanien AE, Azar AT (2017) Feature selection via a novel chaotic crow search algorithm. Neural Comput Appl. https://doi.org/10.1007/s00521-017-2988-6

Sedlaczek K, Eberhard P (2005) Constrained particle swarm optimization of mechanical systems. In: 6th world congresses of structural and multidisciplinary optimization, Rio de Janeiro, Brazil, pp 1–10

Seif Z, Ahmadi MB (2015) An opposition-based algorithm for function optimization. Eng Appl Artif Intell 37:293–306

Shehab A, Elhoseny M, Muhammad K, Sangaiah AK, Yang P, Huang H, Hou G (2018) Secure and robust fragile watermarking scheme for medical images. IEEE Access 6:10269–10278

Shu WH, Shen H (2014) Incremental feature selection based on rough set in dynamic incomplete data. Pattern Recognit 47(12):3890–3906

Suganthan P, Hansen N, Liang J, Deb K, Chen Y, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Nanyang Technological University, Singapore

Tang K, Li X, Suganthan PN, Yang Z, Weise T (2009) Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization. Nature Inspired Computation and Applications Laboratory, Hefei

Tharwat A, Elhoseny M, Hassanien AE, Gabel T, Kumar NA (2018) Intelligent Beziér curve-based path planning model using Chaotic Particle Swarm Optimization algorithm. Cluster Comput. https://doi.org/10.1007/s10586-018-2360-3

Tosserams S, Etman LFP, Rooda JE (2007) An augmented Lagrangian decomposition method for quasi-separable problems in MDO. Struct Multidiscip Optim 34(3):211–227

Xiang L, Wang G (2015) Optimal band selection for hyperspectral data with improved differential evolution. J Ambient Intell Humaniz Comput 6(5):675–688

Xiaohui Y, Elhoseny M, Hamdy KE, Alaa MR (2017) A genetic algorithm-based, dynamic clustering method towards improved WSN longevity. J Netw Syst Manag 25(1):21–46

Xiuyi J, Lin S, Bing Z, Yiyu Y (2016) Generalized attribute reduction in rough set theory. Knowl Based Syst 91:204–218

Xu X, Tang Y, Li J, Hua CC, Guan XP (2015) Dynamic multi-swarm particle swarm optimizer with cooperative learning strategy. Appl Soft Comput 29:169–183

Yang XS (2008) Nature-inspired metaheuristic algorithms. Luniver Press, Bristol

Zhang C, Wang HP (1993) Mixed-discrete nonlinear optimization with simulated annealing. Eng Optim 17(3):263–280