Knowledge-Based Systems 96 (2016) 120–133

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

SCA: A Sine Cosine Algorithm for solving optimization problems

Seyedali Mirjalili[a,b,*]

[a] School of Information and Communication Technology, Griffith University, Nathan Campus, Brisbane, QLD 4111, Australia
[b] Griffith College, Mt Gravatt, Brisbane, QLD 4122, Australia

ARTICLE INFO

Article history:
Received 8 June 2015
Revised 19 December 2015
Accepted 27 December 2015
Available online 6 January 2016

Keywords:
Optimization
Stochastic optimization
Constrained optimization
Meta-heuristic
Population-based algorithm

ABSTRACT

This paper proposes a novel population-based optimization algorithm called Sine Cosine Algorithm (SCA) for solving optimization problems. The SCA creates multiple initial random candidate solutions and requires them to fluctuate outwards or towards the best solution using a mathematical model based on sine and cosine functions. Several random and adaptive variables also are integrated to this algorithm to emphasize exploration and exploitation of the search space in different milestones of optimization. The performance of SCA is benchmarked in three test phases. Firstly, a set of well-known test cases including unimodal, multi-modal, and composite functions are employed to test exploration, exploitation, local optima avoidance, and convergence of SCA. Secondly, several performance metrics (search history, trajectory, average fitness of solutions, and the best solution during optimization) are used to qualitatively observe and confirm the performance of SCA on shifted two-dimensional test functions. Finally, the cross-section of an aircraft's wing is optimized by SCA as a real challenging case study to verify and demonstrate the performance of this algorithm in practice. The results of test functions and performance metrics prove that the algorithm proposed is able to explore different regions of a search space, avoid local optima, converge towards the global optimum, and exploit promising regions of a search space during optimization effectively. The SCA algorithm obtains a smooth shape for the airfoil with a very low drag, which demonstrates that this algorithm can highly be effective in solving real problems with constrained and unknown search spaces. Note that the source codes of the SCA algorithm are publicly available at http://www.alimirjalili.com/SCA.html.

1. Introduction

Optimization refers to the process of finding optimal values for the parameters of a given system from all the possible values to maximize or minimize its output. Optimization problems can be found in all fields of study, which makes the development of optimization techniques essential and an interesting research direction for researchers. Due to the drawbacks of the conventional optimization paradigms, local optima stagnation, and the need to derivate the search space [1], a growing interest has been observed in stochastic optimization approaches [2] over the last two decades [3–5].

Stochastic optimization algorithms consider optimization problems as black boxes [6]. This means that the derivation of the mathematical models is not required because such optimization paradigms only change the inputs and monitor the outputs of the system for maximizing or minimizing its outputs. Another advantage of considering problems as black boxes is the high flexibility, meaning that stochastic algorithms are readily applicable to problems in different fields. As the name of stochastic optimization techniques imply, they optimize optimization problems randomly [7]. Therefore, they intrinsically benefit from higher local optima avoidance compared to the conventional optimization algorithms.

There are different classification for stochastic optimization algorithms in the literature. Two main classifications are based on the inspiration of an algorithm (swarm intelligence-based [8], evolutionary [9], physics-based [10], etc.) and the number of random solutions that an algorithm generates in each step of optimization. The last classification divides the algorithms to two categories: individual-based and population-based algorithms. In the former class, only one solution is generated randomly and improved over the course of optimization. In the latter class, however, an optimization algorithm generates more than one random solution (mostly many) and improves them during optimization.

Due to the above-mentioned advantages, stochastic optimization techniques have become very popular in the literature. This popularity is not only in the field of optimization but also other fields of study. The application of stochastic algorithms can be found in different branches of science and industry. Since the focus

* Tel.: +61 434555738.
E-mail address: ali.mirjalili@gmail.com, seyedali.mirjalili@griffithuni.edu.au

http://dx.doi.org/10.1016/j.knosys.2015.12.022
0950-7051/

of this paper is on the theory, the applications are not discussed further and interested readers are referred to [11,12].

The theoretical researches in the literature can be divided to three main directions: improving the current techniques, hybridizing different algorithms, and proposing new algorithms. In the first approach, researchers try to equip algorithms with different mathematical or stochastic operators to improve the performance. Popular methods in this class are: chaotic maps [13–17], evolutionary operators [18–23], and local searches [24–27]. The second popular research direction deals with hybridizing different algorithms to improve the performance or solve specific problems [28–35]. There is a significant number of hybrid meta-heuristics in the literature such as: PSO-GA [36], PSO-ACO [37], ACO-GA [38], GA-DE [39], PSO-DE [40], ACO-DE [41], KH-CS [42], and KH-BBO [43].

Last but not least, the proposal of new algorithms is a popular research avenue for many researchers. Inspiration of a new algorithm can be from evolutionary phenomena, collective behavior of creatures (swarm intelligence techniques), physical rules, and human-related concepts. Some of the recent and popular algorithms in each of these subclasses are as follows:

- Evolutionary techniques: Genetic Algorithms (GA) [44], Differential Evolution (DE) [45–48], Biogeography-Based Optimization algorithm (BBO) [49], and Evolution Strategy (ES) [50].
- Swarm intelligence techniques: Ant Colony Optimization [51] (ACO), Particle Swarm Optimization (PSO) [52], and Artificial Bee Colony (ABC) algorithm [53].
- Physics-based techniques: Gravitational Search Algorithm (GSA) [54], Colliding Bodies Optimization (CBO) [55], and Black Hole (BH) [56].
- Human-related techniques: League Championship Algorithm (LCA) [57], Mine Blast Algorithm (MBA) [58], and Teaching-Learning-Based Optimization (TLBO) [59].

Despite the significant number of recently proposed algorithms in this field, there is a fundamental questions here as if and why we need more optimization techniques. This question can be answered referring to the so-called No Free Lunch (NFL) theorem [60]. This theorem logically proves that no one can propose an algorithm for solving *all* optimization problems. This means that the success of an algorithm in solving a specific set of problems does not guarantee solving all optimization problems with different type and nature. In other words all the optimization techniques perform equal in average when considering all optimization problems despite the superior performance on a subset of optimization problems. The NFL theorem allows researchers to propose new optimization algorithms or improve/modify the current ones for solving subsets of problems in different fields.

This is also the motivation of this work, in which a simple yet effective optimization algorithm is proposed to optimize real problems with unknown search spaces. The paper also shows that simple mathematical functions can be used to design optimization algorithms in this field. The algorithm proposed utilizes the functions sine and cosine to explore and exploit the space between two solutions in the search space with the hope to find better solutions. It is worth mentioning here that the author has proposed an algorithm called Moth-Flame Algorithm (MFO) [61] recently. The algorithm proposed in this work is completely different in terms of inspiration, mathematical formulation, and real-world application. The MFO algorithm mimics the navigation of moths in nature, whereas the SCA algorithm is based on sine/cosine mathematical functions to solve optimization problems. MFO has been utilized to optimize the shape of a propeller, while SCA is employed to optimize the shape a 2D airfoil in aircraft wings. The rest of the paper is organized as follows:

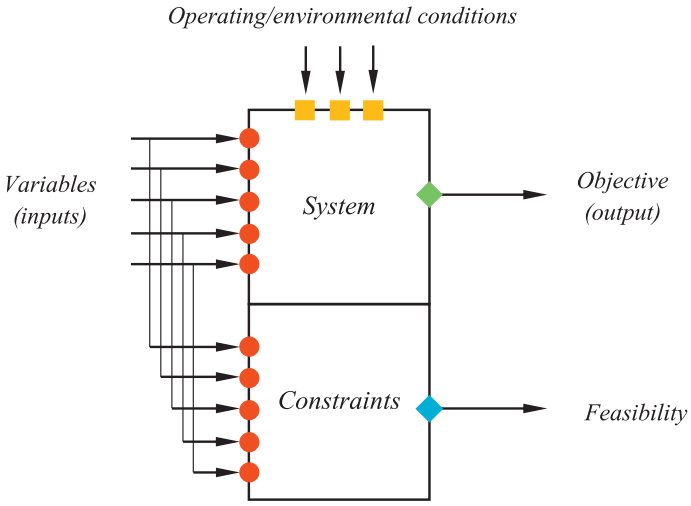


Fig. 1. Different components of an optimization system.

Section 2 includes the preliminaries and essential definitions, presents related works, and reviews the literature. Section 3 demonstrates the mathematical model and proposes the Sine Cosine Algorithm (SCA). The test beds employed and results obtained are presented and discussed in Section 4. The shape of the cross-section of an aircraft's wing is optimized by the SCA algorithm in Section 5, which demonstrates the merits of this algorithm in solving real challenging problems with a large number of constraints and unknown search spaces. Eventually, Section 6 lists the achievement of the paper, concludes the work, and suggests several directions for future studies.

2. Related works

This section first covers the preliminaries and definitions of optimization. The mechanisms and challenges of stochastic/heuristic optimization techniques are then discussed. Eventually, the motivation of this work is provided.

2.1. Preliminaries and definitions

Single-objective optimization deals with optimizing only one objective. This terms stands before multi-objective optimization where there is more than one objective to be optimized. Handling multiple objectives requires special considerations and mechanisms, so the interested readers are referred to the recent review paper witter by Zhou et al. [5] since the focus of this work is on single-objective optimization.

In addition to the objective, other elements involved in the single-objective optimization process are parameters and constraints. Parameters are the variables (unknowns) of optimization problems (systems) that have to be optimized. As Fig. 1 shows, variables can be considered as primary inputs and constraints are the limitations applied to the system. In fact, the constraints define the feasibility of the obtained objective value. Examples of constraints are stress constraints when designing aerodynamic systems or the range of variables.

Other inputs of a system that may affect its output are operating/environmental conditions. Such inputs are considered as secondary inputs that are defined when a system is operating in the simulated/final environment. Examples of such conditions are: temperature/thickness of fluid when a propeller is rotating or the angle of attack when an aircraft is flying. These types of inputs are not optimized by the optimizers but definitely have to be considered during optimization since they may have significant impacts on the outputs.

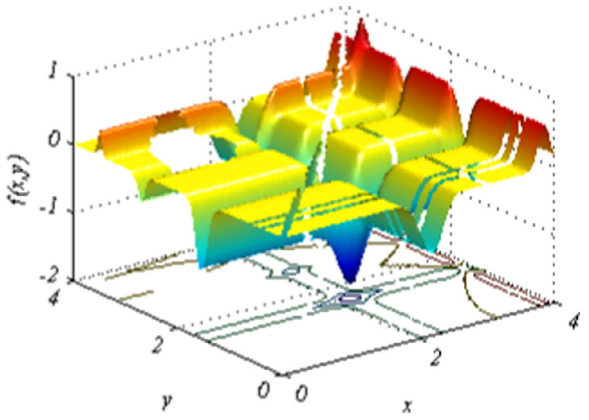


Fig. 2. Example of a search space with two variables and several constraints.

Without the loss of generality, a single-objective optimization can be formulated as a minimization problem as follows:

$$Minimize: \quad f(x_1,\ x_2,\ x_3,\ldots,x_{n-1},x_n) \tag{2.1}$$

$$Subject\ to: \quad g_i(x_1,\ x_2,\ x_3,\ldots,x_{n-1},x_n)\geq 0, \quad i=1,2,\ldots,m \tag{2.2}$$

$$h_i(x_1,\ x_2,\ x_3,\ldots,x_{n-1},x_n)=0, \quad i=1,2,\ldots,p \tag{2.3}$$

$$lb_i\leq x_i\leq ub_i \quad i=1,2,\ldots,n \tag{2.4}$$

where n is number of variables, m indicates the number of inequality constraints, p shows the number of equality constraints, lb_i is the lower bound of the i-th variable, and ub_i is the upper bound of the i-th variable.

As can be seen in Eqs. (2.2) and (2.3), there are two types of constraints: inequality and equality. The set of variables, constraints, and objective constructs a search space for a given problem. Unfortunately, it is usually impossible to draw the search space due to the high-dimensionality of the variables. However, an example of a search space constructed by two variables and several constraints are shown in Fig. 2.

It may be observed in Fig. 2 that the search space can have multiple local optima, but one of them is the global optimum (or more than one in case of a flat landscape). The constraints create gaps in the search space and occasionally split it to various separated regions. In the literature, infeasible regions refer to the areas of the search space that violate constraints.

The search space of a real problem can be super challenging. Some of the difficulties of the real search spaces are discontinuity, large number of local optima, large number of constrains, global optimum located on the boundaries of constraints, deceptive valleys towards local optima, and isolation of the global optimum. An optimization algorithm should be equipped with suitable operators for handling all these difficulties to find the global optimum.

With formulating a problem, an optimizer would be able to tune its variables based on the outputs and constraints. As mentioned in Section 1, one of the advantages of stochastic algorithms is that they consider a system as a black box. Fig. 3 shows that the optimizer only provides the system with variables and observes the outputs. The optimizer then iteratively and stochastically changes the inputs of the system based on the feedbacks (output) obtained so far until the satisfaction of an end criterion. The process of changing the variables based on the history of outputs is defined by the mechanism of an algorithm. For instance, PSO saves the best solutions obtained so far and encourages new solutions to relocate around them.

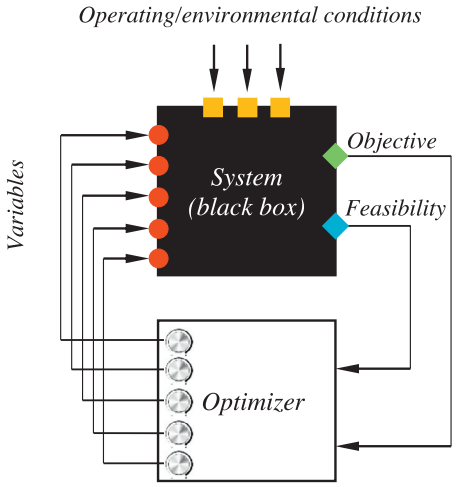


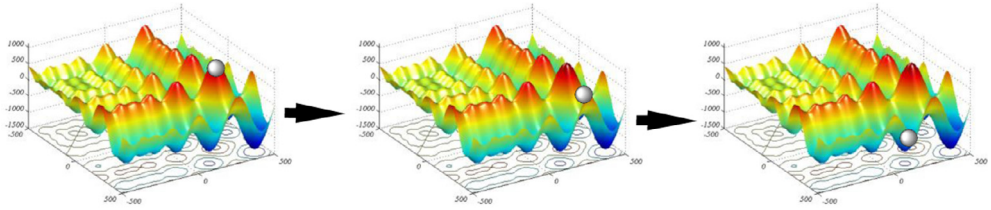
Fig. 3. Stochastic population-based optimizers consider the system as black box.

The literature of stochastic/heuristic optimization techniques and challenges for designing them are reviewed and discussed in detail in the following subsection.

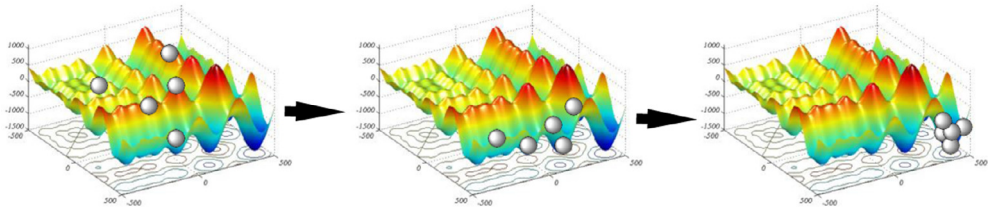
2.2. Literature review

In the field of optimization, in 1977, a revolutionary idea was proposed by Holland where evolutionary concepts in nature was simulated in computer for solving optimization problems [44]. The GA algorithm came to existence and opened a new way of tackling challenging problems in different fields of study. The general idea of the GA algorithm was very simple. It mimicked selection, recombination, and mutation of genes in nature. In fact, the Darwin's theory of evolution was the main inspiration of this algorithm. In GA, the optimization process is started by creating a set of random solutions as candidate solutions (individuals) for a given optimization problem. Each variable of the problem is considered as a gene and the set of variables is analogous to chromosomes. Similarly to nature, a cost function defines the fitness of each chromosome. The whole set of solutions is considered as a population. When the fitness of chromosomes are calculated, the best chromosomes are randomly selected for creating the next population. They main inspiration of the GA algorithm is here, in which the fittest individuals have higher probability to be selected and participated in creating the next population similarly to what is happening in nature. The next step is the combination of the individuals selected. In this step the genes from pairs of individuals are randomly merged to produce new individuals. Eventually, some of the individuals' genes in the population are changed randomly to mimic mutation.

The GA algorithm proved that the nature-inspired paradigms can be very simple yet powerful in optimizing problems. After the proposal of the GA algorithm, the field of stochastic optimization techniques received much attention. The PSO algorithm [52] is the outcome of this popularity several years after the invention of the GA algorithm. The PSO algorithm mimics the social and individual behavior of herd of animals, schools of fishes, or flocks of birds in foraging. Similarly to the GA algorithm, the optimization process starts with a set of randomly created solutions. In addition to the set of solutions, there is another set called velocity set which is responsible for storing and defining the amount of movement of particles. During optimization, the velocity of a particle is updated based on the best solution that it has obtained so far as well as the best solution that the swarm has found. There are three random components in defining the tendency towards previous velocity, effect of the personal best, and the impact of the global best.



(a) Individual-based stochastic optimization



(b) Population-based stochastic optimization

Fig. 4. Individual-based versus population-based stochastic optimization algorithms.

Since the best solutions are saved in the PSO algorithm, there is always high possibility of finding better solutions when searching around them. This is the key reason about the success of the PSO algorithm.

After the development of GA and PSO algorithms, several algorithms were developed and proposed as well. As mentioned in the introduction, they can be divided to two main classes: individual-based versus population-based algorithms. The individual-based algorithm creates only a single solutions and evolves/improves it over the course of iterations. However, a population-based algorithm initializes the optimization process by more than one solutions. The solutions in this set are then enhanced over the course of iterations. The way that these two families perform optimization are illustrated in Fig. 4. The advantage of individual-based algorithms is the need for a low number of function evaluation because a single solution only needs one function evaluation. Therefore, such optimization techniques require $1 \times T$ number of function evaluations where T is the maximum number of iterations. However, high probability of local optima stagnation and lack of information sharing are the main drawbacks of these algorithms, which is due to the low number of solutions. Fig. 4(a) shows that the single candidate solution entraps in the local optima which is very close the global optimum.

In contrary, population-based algorithms benefit from high local optima avoidance since they employ multiple solutions. Fig. 4(b) illustrates how the collection of candidate solutions results in finding the global optimum. Multiple solutions also assist a population-based algorithm to collect information from different regions of the search space easily. This is done by information exchange between the search agents during the optimization process. Therefore, search agents are able to better and faster explore and exploit search spaces. However, the main drawbacks of these methods is the large number of function evaluation. Such optimization techniques require $n \times T$ number of function evaluations where n is the number of solutions (search agents) and T is the maximum number of iterations.

2.3. Motivation of this work

Despite the need for more function evaluations, the literature shows that population-based algorithms are highly suitable for solving real challenging problems since they are able avoid local optima, explore the search space, and exploit the global optimum more reliably compared to individual-based algorithms. In addition, the NFL theorem says that all algorithms perform equal on all optimization problems. Therefore, there are still problems that have not yet been solved, or they can be solved better by new algorithms. These two reasons are the main motivations of this work, in which a novel population-based optimization algorithm is proposed and compared to the current well-known algorithms in the literature.

3. Sine Cosine Algorithm (SCA)

Generally speaking, population-based optimization techniques start the optimization process with a set of random solutions. This random set is evaluated repeatedly by an objective function and improved by a set of rules that is the core of an optimization technique. Since population-based optimization techniques look for the optima of optimization problems stochastically, there is no guarantee of finding a solution in a single run. However, with enough number of random solutions and optimization steps (iterations), the probability of finding the global optimum increases.

Regardless of the differences between algorithms in the field of stochastic population-based optimization, the common is the division of optimization process to two phases: exploration versus exploitation [62]. In the former phase, an optimization algorithm combines the random solutions in the set of solutions abruptly with a high rate of randomness to find the promising regions of the search space. In the exploitation phase, however, there are gradual changes in the random solutions, and random variations are considerably less than those in the exploration phase.

In this work, the following position updating equations are proposed for both phases:

$$X_i^{t+1} = X_i^t + r_1 \times \sin(r_2) \times \left| r_3 P_i^t - X_i^t \right| \tag{3.1}$$

$$X_i^{t+1} = X_i^t + r_1 \times cos(r_2) \times \left| r_3 P_i^t - X_{i\,i}^t \right| \tag{3.2}$$

where X_i^t is the position of the current solution in i-th dimension at t-th iteration, $r_1/r_2/r_3$ are random numbers, P_i is position of the destination point in i-th dimension, and || indicates the absolute value.

These two equations are combined to be used as follows:

$$X_i^{t+1} = \begin{cases} X_i^t + r_1 \times \sin(r_2) \times \left| r_3 P_i^t - X_i^t \right|, & r_4 < 0.5 \\ X_i^t + r_1 \times cos(r_2) \times \left| r_3 P_i^t - X_{i\,i}^t \right|, & r_4 \geq 0.5 \end{cases} \tag{3.3}$$

where $r4$ is a random number in [0,1]

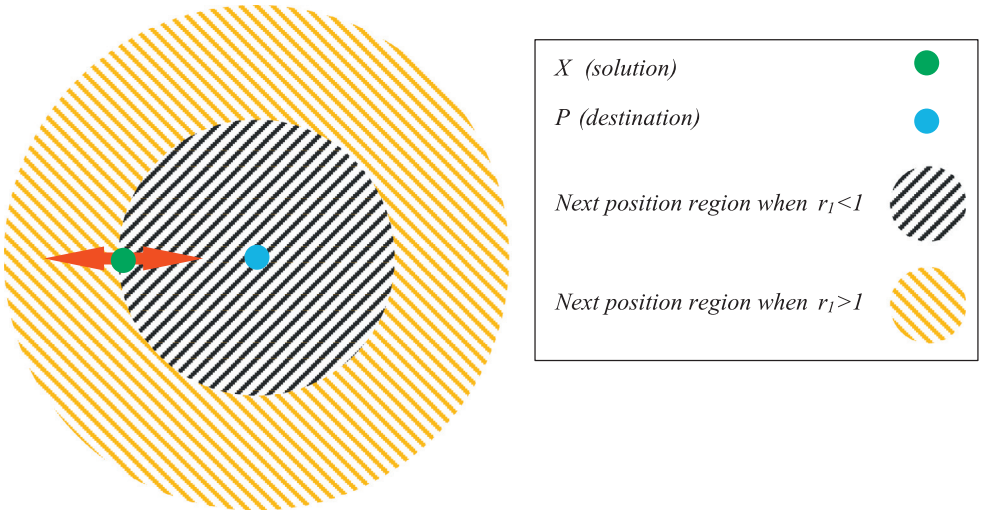


Fig. 5. Effects of Sine and Cosine inn Eqs. (3.1) and (3.2) on the next position.

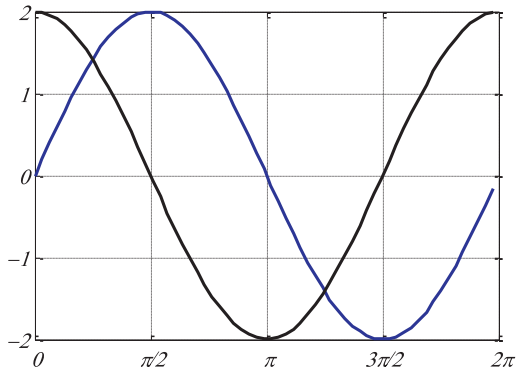


Fig. 6. Sine and cosine with range of [−2,2].

As the above equations show, there are four main parameters in SCA: r_1, r_2, r_3, and r_4. The parameter r_1 dictates the next position's region (or movement direction) which could be either in the space between the solution and destination or outside it. The parameter r_2 defines how far the movement should be towards or outwards the destination. The parameter r_3 brings a random weight for the destination in order to stochastically emphasize ($r_3 > 1$) or deemphasize ($r_3 < 1$) the effect of destination in defining the distance. Finally, the parameter r_4 equally switches between the sine and cosine components in Eq. (3.3).

Due to the use of sine and cosine in this formulation, this algorithm is name Sine Cosine Algorithm (SCA). The effects of Sine and Cosine on Eqs. (3.1) and (3.2) are illustrated in Fig. 5. This figure shows that how the proposed equations define a space between two solutions in the search space. It should be noted that this equation can be extended to higher dimensions although a two-dimensional model is illustrated in Fig. 5. The cyclic pattern of sine and cosine function allows a solution to be re-positioned around another solution. This can guarantee exploitation of the space defined between two solutions. For exploring the search space, the solutions should be able to search outside the space between their corresponding destinations as well. This can be achieved by changing the range of the sine and cosine functions as shown in Fig. 6.

A conceptual model of the effects of the sine and cosine functions with the range in [−2, 2] is illustrated in the Fig. 7. This figure shows how changing the range of sine and cosine functions requires a solution to update its position outside or inside the space between itself and another solution. The random location either inside or outside is achieved by defining a random number for r_2 in [0, 2π] in Eq. (3.3). Therefore, this mechanism guarantees exploration and exploitation of the search space respectively.

An algorithm should be able to balance exploration and exploitation to find the promising regions of the search space and eventually converge to the global optimum. In order to balance exploration and exploitation, the range of sine and cosine in Eqs. (3.1) to (3.3) is changed adaptively using the following equation:

$$r_1 = a - t\frac{a}{T} \tag{3.4}$$

where t is the current iteration, T is the maximum number of iterations, and a is a constant.

Fig. 8 shows how this equation decreases the range of sine and cosine functions over the course of iterations. It may be inferred from Figs. 7 and 8 that the SCA algorithm explores the search space when the ranges of sine and cosine functions are in (1,2] and [−2, −1). However, this algorithm exploits the search space when the ranges are in the interval of [−1,1].

After all, the pseudo code of the SCA algorithm is presented in Fig. 9. This figure shows that the SCA algorithm starts the optimization process by a set of random solutions. The algorithm then saves the best solutions obtained so far, assigns it as the destination point, and updates other solutions with respect to it. Meanwhile, the ranges of sine and cosine functions are updated to emphasize exploitation of the search space as the iteration counter increases. The SCA algorithm terminates the optimization process when the iteration counter goes higher than the maximum number of iterations by default. However, any other termination condition can be considered such as maximum number of function evaluation or the accuracy of the global optimum obtained.

With the above operators, the proposed algorithm theoretically is able to determine the global optimum of optimization problems due to the following reasons:

- SCA creates and improves a set of random solutions for a given problem, so it intrinsically benefits from high exploration and local optima avoidance compared to individual-based algorithms.
- Different regions of the search space are explored when the sine and cosine functions return a value greater than 1 or less than −1.
- Promising regions of the search space is exploited when sine and cosine return value between −1 and 1.

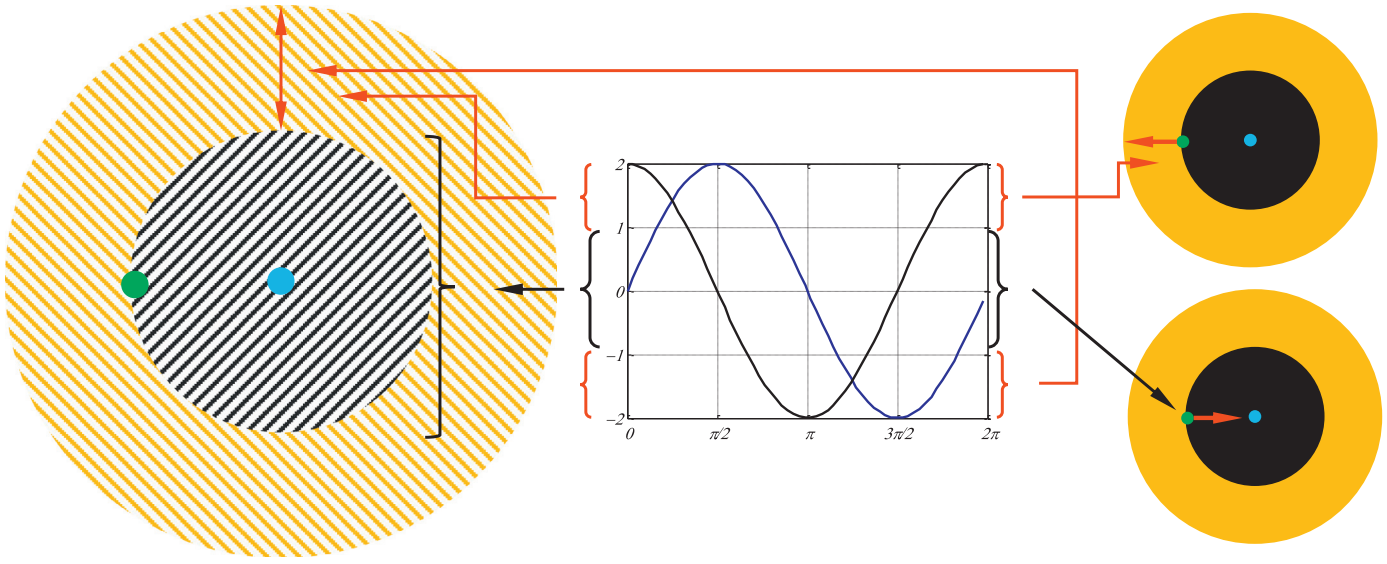


Fig. 7. Sine and cosine with the range in [−2,2] allow a solution to go around (inside the space between them) or beyond (outside the space between them) the destination.

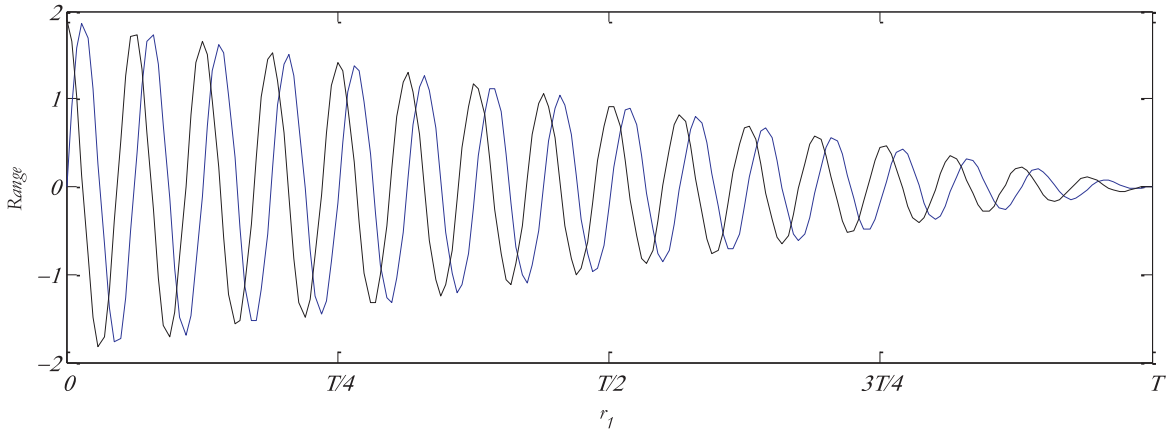


Fig. 8. Decreasing pattern for range of sine and cosine ($a = 3$).

```
Initialize a set of search agents (solutions) (X)
Do
   Evaluate each of the search agents by the objective function
   Update the best solution obtained so far (P=X*)
   Update r1, r2, r3, and r4
   Update the position of search agents using Eq. (3.3)
While(t< maximum number of iterations)
Return the best solution obtained so far as the global optimum
```

Fig. 9. General steps of the SCA Algorithm.

- The SCA algorithm smoothly transits from exploration to exploitation using adaptive range in the sine and cosine functions.
- The best approximation of the global optimum is stored in a variable as the destination point and never get lost during optimization.
- Since the solutions always update their positions around the best solution obtained so far, there is a tendency towards the best regions of the search spaces during optimization.
- Since the proposed algorithm considers optimization problem as black boxes, it is readily incorporable to problems in different fields subject to proper problem formulation.

The next sections employ a wide range of test problems and one real case study to investigate, analyse, and confirm the effectiveness of the SCA algorithm.

4. Results and discussion

In the field of optimization using meta-heuristics and evolutionary algorithms, several test cases should be employed to confirm the performance of an algorithm. This is due to the stochastic nature of these algorithms, in which a proper and sufficient set of test functions and case studies should be employed to confidently make sure that the superior results are not happened by chance. However, there is no clear definition of suitability for a set of benchmark cases studies. Therefore, researchers try to test their algorithms on as many test cases as possible. This paper also employs several test functions with different characteristics. Later, a real challenging Computational Fluid Dynamics (CFD) problem is solved by the SCA algorithm as well.

The set of cases studies employed includes three families of test functions: unimodal, multi-modal, and composite test

Table 1
Results on benchmark functions.

F	SCA		PSO		GA		BA		FPA		FA		GSA
	ave	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*	*std*	*ave*
F1	0.0000	0.0000	0.0003	0.0011	0.8078	0.4393	1.0000	1.0000	0.2111	0.0717	0.0004	0.0002	0.0000
F2	0.0000	0.0001	0.0693	0.2164	0.5406	0.2363	1.0000	1.0000	0.9190	0.7804	0.0177	0.0179	0.0100
F3	0.0371	0.1372	0.0157	0.0158	0.5323	0.2423	1.0000	1.0000	0.2016	0.1225	0.0000	0.0004	0.0016
F4	0.0965	0.5823	0.0936	0.4282	0.8837	0.7528	1.0000	1.0000	0.8160	0.5618	0.0000	0.0107	0.1177
F5	0.0005	0.0017	0.0000	0.0000	0.6677	0.4334	1.0000	1.0000	0.0813	0.0426	0.0000	0.0000	0.0000
F6	0.0002	0.0001	0.0004	0.0033	0.7618	0.7443	1.0000	1.0000	0.2168	0.1742	0.0004	0.0002	0.0000
F7	0.0000	0.0014	0.0398	0.0634	0.5080	0.1125	1.0000	1.0000	0.3587	0.2104	0.0009	0.0022	0.0021
F8	1.0000	0.0036	1.0000	0.0036	1.0000	0.0055	0.0000	1.0000	1.0000	0.0029	1.0000	0.0168	1.0000
F9	0.0000	0.7303	0.3582	0.8795	1.0000	0.6881	0.4248	1.0000	0.8714	0.8665	0.0190	0.3298	0.0222
F10	0.3804	1.0000	0.1045	0.0541	0.8323	0.0686	0.8205	0.0796	1.0000	0.0162	0.0000	0.0079	0.1569
F11	0.0000	0.0051	0.0521	0.0448	0.7679	0.2776	1.0000	1.0000	0.2678	0.0706	0.0074	0.0001	0.4011
F12	0.0000	0.0000	0.0000	0.0000	0.4573	0.4222	1.0000	1.0000	0.0008	0.0015	0.0000	0.0000	0.0000
F13	0.0000	0.0000	0.0000	0.0000	0.6554	0.8209	1.0000	1.0000	0.0187	0.0375	0.0000	0.0000	0.0000
F14	0.3908	0.1924	0.1816	1.0000	0.4201	0.1610	1.0000	0.6977	0.3786	0.1716	0.0000	0.9571	0.0961
F15	0.0230	0.0676	0.3016	1.0000	0.0000	0.0779	1.0000	0.7614	0.2235	0.4252	0.4395	0.9135	0.2926
F16	0.0497	0.4921	0.0427	0.7228	0.0000	0.2422	0.3572	0.7629	0.2652	0.6012	0.5298	1.0000	1.0000
F17	0.0000	0.1105	0.0249	1.0000	0.1093	0.1873	0.8189	0.7754	0.5197	0.4847	0.7093	0.8842	0.7887
F18	0.0129	0.0134	0.1772	0.4289	0.0000	0.0538	1.0000	0.2855	0.1310	0.0429	0.0723	0.2069	0.8018
F19	0.0000	0.2001	0.7727	1.0000	0.0192	0.0312	1.0000	0.2142	0.3192	0.4635	0.8176	0.7924	0.9950
Sum	1.9911	3.5379	3.2346	6.8619	9.9634	5.9972	16.4214	15.5767	7.8004	5.1479	3.6143	5.1403	5.6858

Table 2
p-Values of the Wilcoxon ranksum test over all runs ($p >= 0.05$ have been underlined).

F	SCA	PSO	GA	BA	FPA	FA	GSA
F1	N/A	0.002165	0.002165	0.002165	0.002165	0.002165	0.002165
F2	N/A	0.002165	0.002165	0.002165	0.002165	0.002165	0.002165
F3	0.004329	0.002165	0.002165	0.002165	0.002165	N/A	0.008658
F4	0.002165	0.002165	0.002165	0.002165	0.002165	N/A	0.002165
F5	N/A	0.002165	0.002165	0.002165	0.002165	0.002165	0.681818
F6	0.002165	0.002165	0.002165	0.002165	0.002165	0.002165	N/A
F7	N/A	0.002165	0.002165	0.002165	0.002165	0.24026	0.002165
F8	0.002165	0.002165	0.002165	N/A	0.002165	0.002165	0.002165
F9	N/A	0.002165	0.002165	0.002165	0.002165	0.484848	0.818182
F10	1.000000	0.002165	0.002165	0.002165	0.002165	N/A	0.093074
F11	N/A	0.002165	0.002165	0.002165	0.002165	0.002165	0.002165
F12	N/A	0.015152	0.002165	0.002165	0.002165	0.064935	0.064935
F13	0.002165	0.002165	0.002165	0.002165	0.002165	N/A	0.393939
F14	0.064935	0.588745	0.064935	0.041126	0.064935	N/A	0.132035
F15	0.179654	0.064935	N/A	0.002165	0.008658	0.008658	0.002165
F16	0.818182	0.937229	N/A	0.002165	0.002165	0.002165	0.002165
F17	N/A	1.000000	0.015152	0.002165	0.002165	0.002165	0.002165
F18	0.818182	0.393939	N/A	0.002165	0.002165	0.699134	0.025974
F19	N/A	0.064935	0.699134	0.002165	0.041126	0.041126	0.002165

functions [63–66]. The mathematical formulation of these test functions are available in the appendix. The first family of test functions has no local optima and there is only one global optima. This makes them highly suitable for testing the convergence speed and exploitation of algorithms. The second group of test functions, however, has multiple local solutions in addition to the global optimum. These characteristics are beneficial for testing local optima avoidance and explorative ability of an algorithm. Finally, the composite test functions are the rotated, shifted, biased, and combined version of several unimodal and multi-modal test functions.

For solving the aforementioned test functions, a total of 30 search agents are allowed to determine the global optimum over 500 iterations. The SCA algorithm is compared to Firefly Algorithm (FA) [67], Bat Algorithm (BA) [68], Flower Pollination Algorithm (FPA) [69], Gravitational Search Algorithm (GSA) [54], PSO and GA for verification of the results. Since the results of a single run might be unreliable due to the stochastic nature of meta-heuristics, all of the algorithms are run 30 times and statistical results (mean and standard deviation) are collected and reported in Table 1. Note that the results are normalized in [0, 1] to compare the results of all test functions. To decide about the significance of the results, a non-parametric statistical test called Wilcoxon ranksum test is conducted as well. The p-values obtained from this statistical test are reported in Table 2.

The results in Table 1 show that the SCA algorithm outperforms others on the majority of the test cases. Firstly, the SCA algorithm shows superior results on 3 out of 6 unimodal test functions. The p-values in Table 2 show that this superiority is statistically significant. Due to the characteristics of the unimodal test functions, these results strongly show that the SCA algorithm has a high exploitation and convergence. Secondly, Table 1 shows that the SCA algorithm outperforms all of the algorithms employed on the majority of the multi-modal test functions (F7, F9, F11, and F12). The p-values in Table 2 also support the better results of SCA statistically. Inspecting the results of this table, the SCA algorithm provides p-values greater than 0.05 for the rest of test functions, showing that this algorithm is very competitive. These results prove that the SCA algorithm benefits from high exploration and local optima avoidance. Finally, the results of the proposed algorithm on the composite test functions in Tables 1 and 2 demonstrate the merits of SCA in solving problems with challenging search spaces. Due to the normalization of the results, the

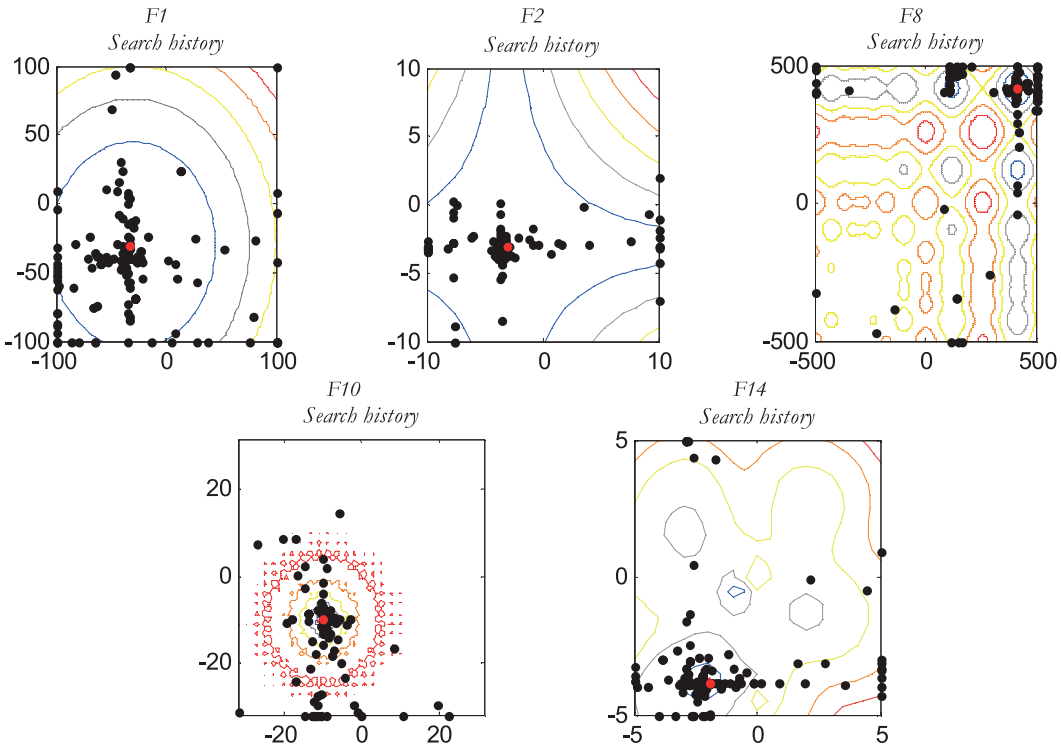


Fig. 10. Search history of search agents when solving the test problems.

overall performance of algorithms can be compared as well. The last row of Table 1 presents the summation of the average and standard deviation of algorithms on all test functions. It is evident that SCA shows the minimum values for both *ave* and *std*, proving that this algorithm reliably outperforms others in total.

Although the above-discussed results prove and verify the high performance of the SCA algorithm, there are several other experiments that need to be done to confidently confirm the performance of this algorithm in solving real problems. In other words, the behavior of search agents during optimization should be monitored to observe: how they move around the search space, if they face abrupt changes in the initial stages of optimization to explore the search space, if they undergo small changes in the final steps of iteration to exploit the search space, how they converge towards the promising regions of the search space, how they improve their initial random solutions, and how they improve their fitness values over the course of iterations. In order to observe the behaviour of search agents, the two-dimensional version of the test functions is solved by 4 search agents. Note that the optima of some of the test functions are shifted to locations other than the origin to provide more challenging test beds. The search history of the search agents is illustrated in Fig. 10. This figure shows that the SCA algorithm searches around the promising regions of the search space. The distribution of the sampled points around the global optima is substantially high, which shows that the SCA algorithm exploits the most promising region of the search space in addition to the exploration. However, it is not clear from this figure if the search agents first start exploration or exploitation. To observe this, Fig. 11 is provided in this regard, which illustrates the fluctuations of the first dimension in the first search agent.

Fig. 11 shows that the search agents face abrupt fluctuations in the early steps of optimization. However, the sudden changes are decreased gradually over the course of iterations. This confirms that the search agents first explore the search space and then converge around the best solution obtained in the exploration phase. There is a question here as how to make sure that all of the search agents are improved during optimization despite the rapid and steady changes in Fig. 11. In order to confirm the improvement of all solutions, the average fitness of all search agents during optimization is illustrated in Fig. 12.

This figure shows that the average fitness of all search agents tend to be decreased over the course of iterations. The interesting pattern that can be observed in this figure is the high fluctuation of the average fitness in the exploration phase (until nearly the 50th iteration) and low changes in the average fitness in the exploitation phase (after 50th iteration). Deterioration of the fitness of some of the search agents is unavoidable in the exploration phase where the SCA algorithm should discover the promising regions of the search space. However, the observed patterns in Fig. 12 show that the fitness of search agents has a descending behavior over the course of iterations. This proves that the SCA algorithm is able to eventually improve the fitness of initial random solutions for a given optimization problem.

In the previous paragraphs, it was claimed that the search agents of the SCA algorithm tend to explore the promising regions of the search space and exploit the best one eventually. However, the convergence behavior of the algorithm was not observed and verified. Although this can be inferred indirectly from the trajectory and average fitness, the convergence curves of SCA are depicted in Fig. 13.

This figure illustrates the best solution obtained so far during optimization. The descending trend is quite evident in the convergence curves of SCA on all of the test functions investigated. This strongly evidences the ability of the SCA algorithm in obtaining a better approximation of the global optimum over the course of iterations.

The results and discussions in this section prove that the SCA algorithm proposed is able to determine the global optima of the test functions. Although it can be claimed here that this algorithm would be able to approximate the global optima of real problems, there is a main difference between real problems and benchmark functions. The shape of search space and the location

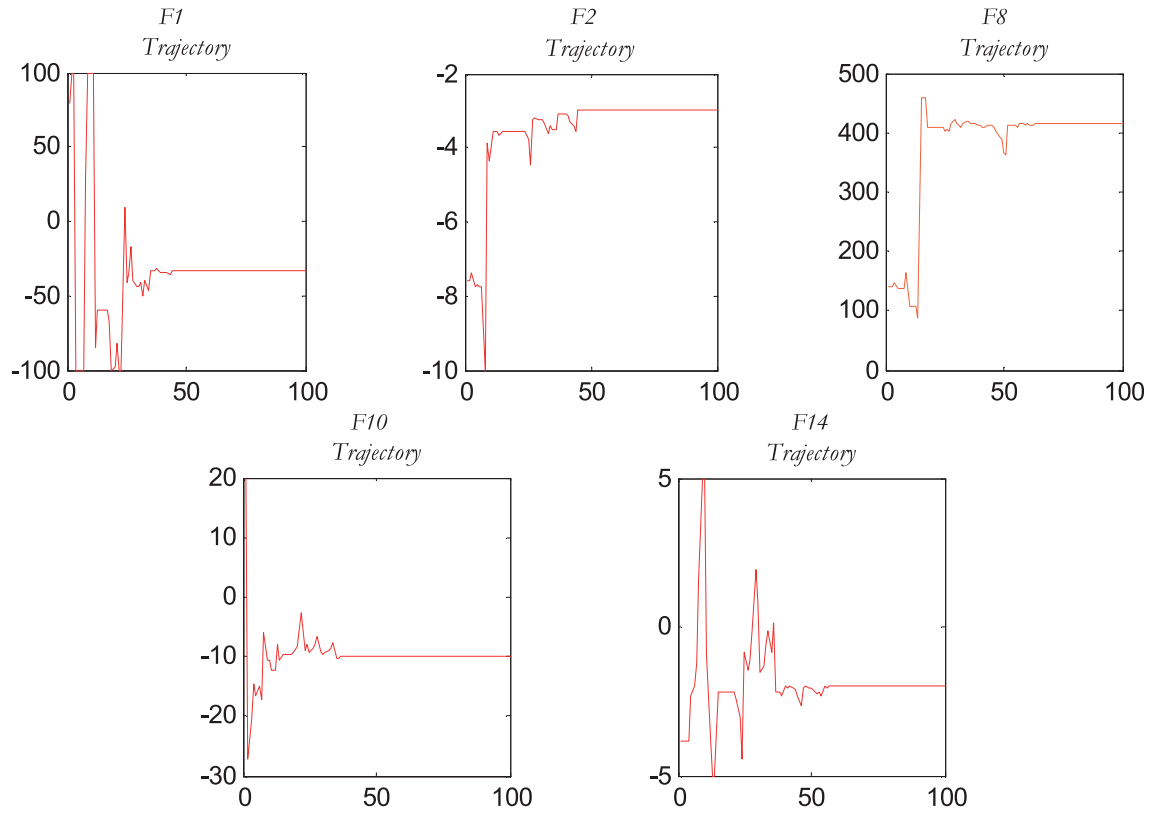


Fig. 11. Trajectory of the first variable of the first search agent when solving the test problems.

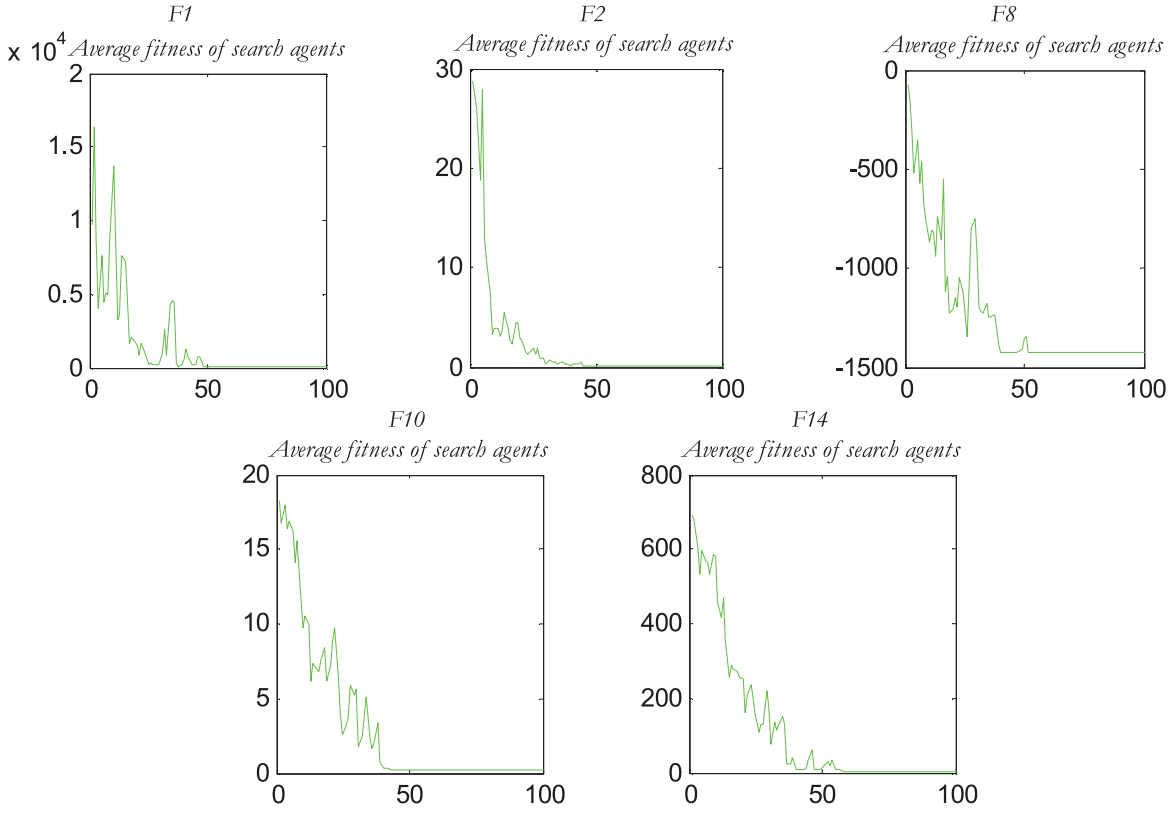


Fig. 12. Average fitness of search agents during optimization.

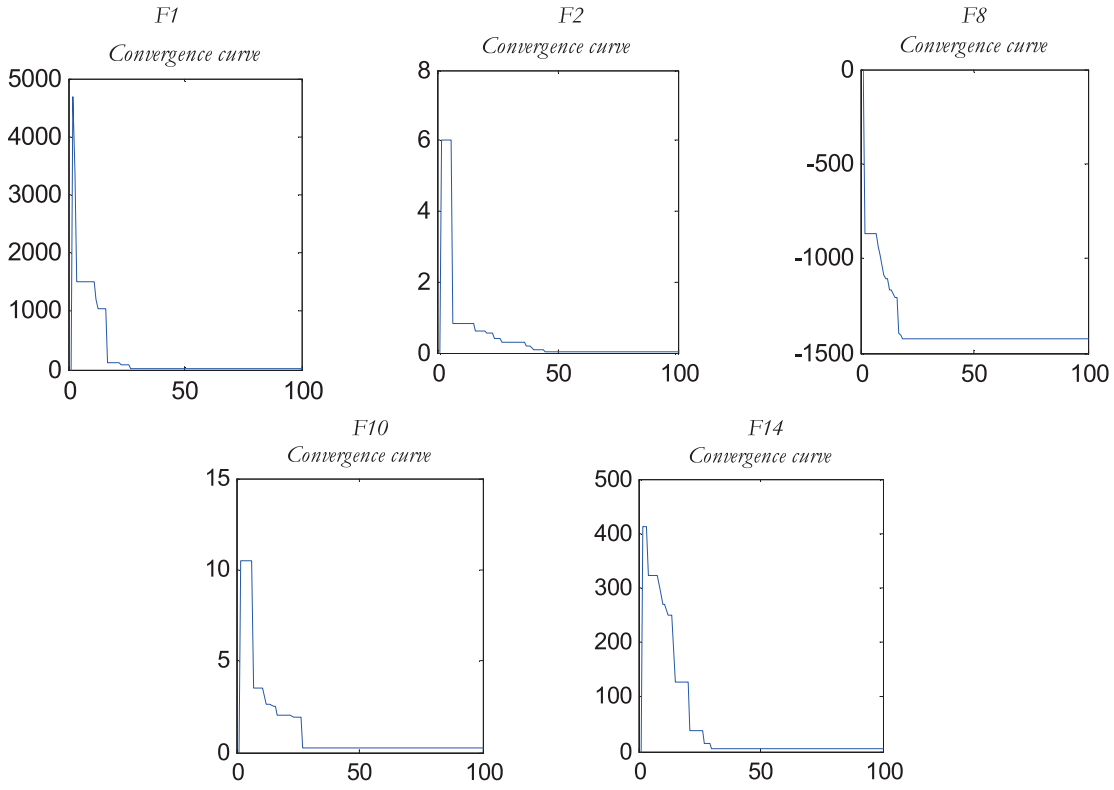


Fig. 13. Convergence curve (best solution in each iteration) of the SCA algorithm.

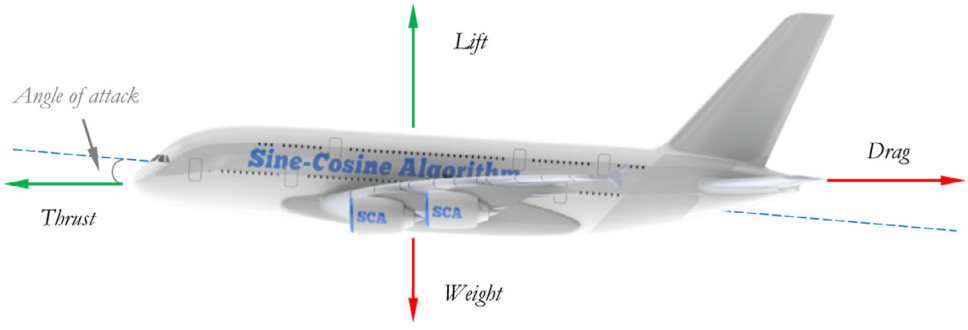


Fig. 14. Different forces that apply to an airplane.

of the global optimum of the test functions are known, while those of real problems are mostly unknown. In addition, the real problems are accompanied by a large number of equality and inequality constraints. Therefore, there is a need to investigate the performance of the SCA algorithm in solving at least one real challenging constrained problem with unknown global optimum and search space. This is the motivation of the next section, in which the two-dimensional cross-section of an aircraft's wing is optimized by SCA to confirm its performance in practice.

5. Airfoil design using SCA

The problem investigated in this subsection is airfoil design. There are two objectives in this problem: lift versus drag. There two forces are shown in Fig. 14. It may be observed that lift is when the thrust force is converted to a vertical force, which causes flying of a plane. However, drag is the opposite force that is applied to the wing and causes decreasing speed. The lift and drag are in conflict, meaning that increasing one results in decreasing the other. In a real airplane both of these forces are desirable in different occasions. When the airplane is taking off, ascending, and cruising maximum lift and minimum drag is fruitful. When descending, landing, and touching down the drag becomes important to slow down the speed of the vehicle. In this section the drag is only considered, so the main objective is to minimize this force. In other words, this section employs the SCA algorithm to define the best shape for the wing to minimize drag.

To design an aircraft wing, several components should be considered: shape of the cross section of the wing (airfoil), the overall shape of the wing, flaps, internal frames, and position of engines. This paper only concentrates on designing a 2D airfoild, which is the main and essential component in a wing. The shape of a 2D airfoil is illustrated in Fig. 15.

There are different version of this problem in the literature in terms of the design parameters. In this work, the B-spline is utilized to define the shape of the airfoil. As shown in Fig. 16, there are eight controlling parameters of which one of the leading points is fixed. The rest of controlling parameters, however, are allowed

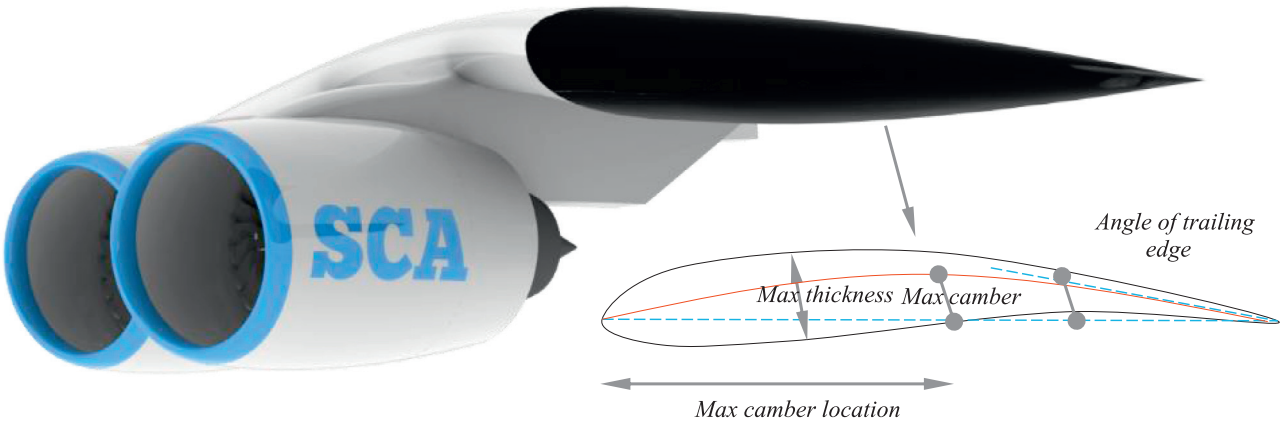


Fig. 15. Cross section of a real with a 2D airfoil.

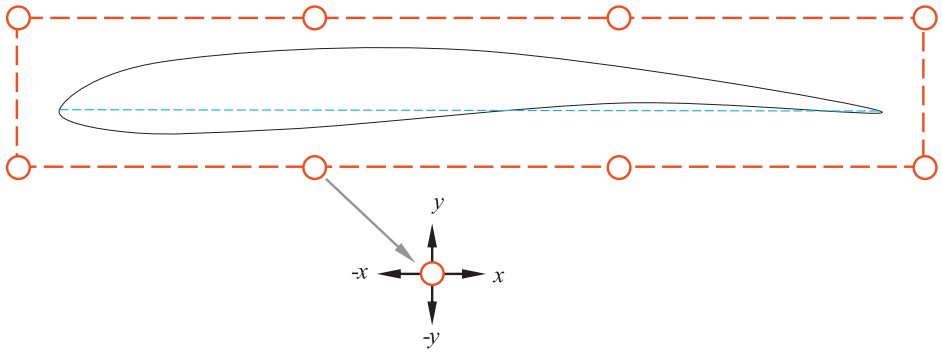


Fig. 16. B-spline for the problem of Airfoil design.

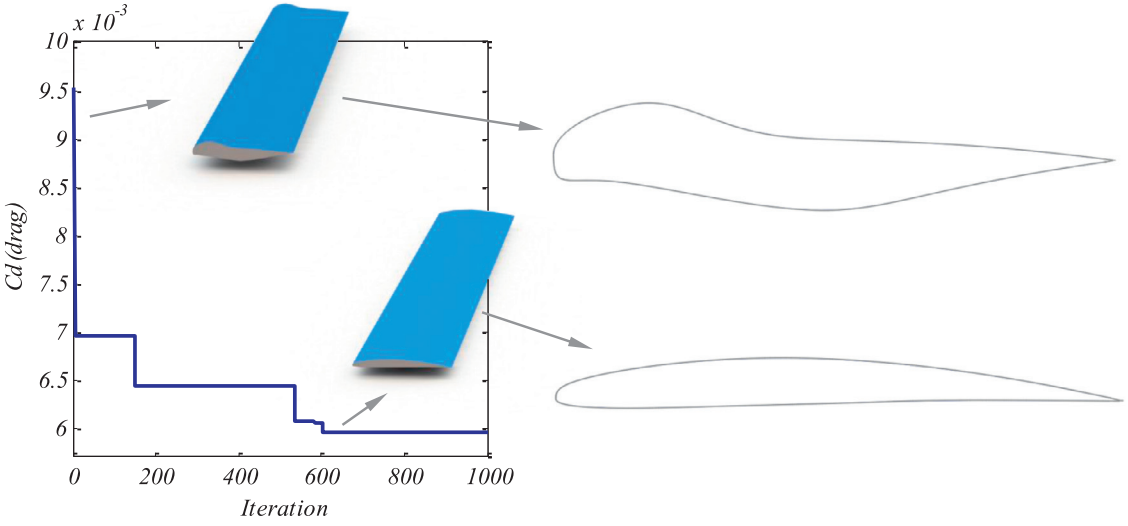


Fig. 17. Convergence curve of the SCA on the airfoil design problem, initial aifoil, and optimized airfoil.

to move along both directions of x and y axes. Therefore, there is a total of 14 (7×2) parameters, which are the x and y positions of the seven controlling points. The problem of airfoil design is formulated for the SCA algorithm as follows:

$$\begin{aligned} &\textit{Minimize}: \quad F(\vec{x},\ \vec{y}) = C_d(\vec{x},\ \vec{y}) \\ &\textit{Subject to}: \quad -1 \le \vec{x},\ \vec{y} \le 1,\ \textit{satisfaction of CO set} \end{aligned} \tag{5.1}$$

where $\vec{x} = \{x_1,\ x_2, \ldots, x_7\}$, $\vec{y} = \{y_1, y_2,\ \ldots, y_7)\}$, CO includes many constraints such as minimum of thickness, maximum of thickness, etc.

A freeware called XFoil is used for calculating drag [75]. It may be seen in Eq. (5.1) that the problem is subject to several constraints. Generally speaking, Computational Fluid Dynamics (CFD) problem are highly constrained, which make them very challenging. For solving such problems, an optimization algorithm should be equipped with a proper constraint handling method. There are different approaches in the literature to cope with constraints of which penalty functions are the simplest ones. In such methods, the main objective function is penalized by a penalty function with respect to the level of constraints' violation. Other powerful

constraint handling methods can be found in [70–73]. Interested readers are referred to the comprehensive literature review by Coello Coello [74]. In this work the following penalty function is utilized, which penalizes F proportional to the level of violation:

$$F(\vec{x},\ \vec{y}) = F(\vec{x},\ \vec{y}) + p\sum_{i=1}^{3} P_i \tag{5.2}$$

where p is a constant and P_i is the violation size on the *i-th* constraint in the *CO* set in Eq. (5.1).

For solving this problem, 30 search agents is employed and allowed to determine the optimal shape for the airfoil over 1000 iterations. The algorithm is run 4 times and the best results are illustrated in Fig. 17.

This figure clearly shows that the SCA algorithm improves the initial random shape for the airfoil to minimize drag. The improvement is quite significant, in which drag was reduced from 0.009 to 0.0061. These results highly demonstrate that the SCA algorithm is able to solve real problems with unknown, challenging, and constrained search spaces. This is due to several reasons. Firstly, SCA is a population-based algorithm, so it intrinsically benefits from high exploration and local optima avoidance. This assists the SCA algorithm to avoid the large number of local solutions in a real search space and explore different regions extensively. Secondly, SCA smoothly transits from exploration to exploitation using the adaptive mechanism for the range of since and cosine functions. This causes local optima avoidance at the beginning of optimization and quick convergence towards the most promising region of the search space in the final steps of optimization. Thirdly, SCA obliges the solutions to update their positions around the best solution obtained so far as the destination point. Therefore, there is always a tendency towards the best regions of the search spaces during optimization and chances for improving the solutions are considerably high. Finally, the SCA algorithm considers optimization problems as black boxes, so it is readily incorporable to problems in different fields subject to the proper formulation of the problem. In addition, the problem independency allows this algorithm not to need gradient information of the search space and work with any types of penalty functions for solving constrained problems.

6. Conclusion

In this paper a novel population-based optimization algorithm was proposed as an alternative for solving optimization problems among the current techniques in the literature. In the SCA algorithm proposed, the solutions were required to update their positions with respect to the best solution obtained so far as the destination point. The mathematical model of position updating fluctuated the solutions outwards or towards the destination point to guarantee exploration and exploitation of the search space, respectively. Several random and adaptive variables also facilitated divergence and convergence of the search agents in the SCA algorithm. To benchmark the performance of SCA, several experiments were done. Firstly, the a set of well-known test cases including unimodal, multi-modal, and composite functions were employed to test exploration, exploitation, local optima avoidance, and convergence of the proposed algorithm. Secondly, the two-dimensional versions of some of the test functions were chosen and re-solved by SCA. Several performance metrics (search history, trajectory, average fitness of solutions, and best solution during optimization) were employed to qualitatively observe and confirm the performance of SCA. Finally, the shape of a two-dimensional airfoil (cross-section of an aircraft's wing) was optimized by SCA as a real challenging case study to verify and demonstrate the performance of this algorithm in solving real problems with constrained and unknown search spaces.

The results of unimodal test functions showed that the SCA algorithm converged substantially faster than FA, BA, FPA, GSA, PSO and GA. A similar behavior was observed in the multi-modal test functions, which proved the high exploration and local optima avoidance of the algorithm proposed. As per the results of composite test functions, SCA outperformed other algorithms occasionally, which showed that this algorithm was also able to successfully balance exploration and exploitation to determine the global optima of challenging test functions. The results of performance metrics proved that SCA required its search agent to change abruptly in the initial stage of optimization and gradually in the final steps of optimization. The results showed that this behavior caused exploration of the search space extensively and exploitation of the most promising region. The average fitness of solutions and convergence curves also evidenced and confirmed the improvement of initial random population and the best solution obtained so-far by SCA. The results of the first two test phases proved the SCA is able to successfully solve test problems, which have known shape of search space. The results of SCA on the aroifoil design problem also showed that this algorithm have the potential to solve challenging real problems as well. The airfoil design problem was a highly constrained case study with a completely unknown search space. Therefore, the results of real case study highly demonstrated and confirmed the merits of SCA in solving real problems.

As per the findings of this paper and referring the NFL theorem, it can be concluded that the SCA can be a very suitable alternative compared the current algorithms in the literature for solving different optimization problems. On the other hand, this algorithm might not be able to outperform other algorithms on specific set of problems, but definitely worth testing and applying to problems in different fields. Therefore, the SCA algorithm is offered to researchers in different fields. The source codes of this algorithm are publicly available at http://www.alimirjalili.com/SCA.html.

This paper opens up several research directions for future studies. Firstly, binary and multi-objective version of this algorithm can be proposed to solve problems with binary and multiple objectives respectively. Secondly, levy flight, mutation, and other evolutionary operators can be integrated to this algorithm for improving its performance. Thirdly, the SCA algorithm can be hybridized with other algorithms in the field of stochastic optimization to improve its performance. Finally, investigation of the application of SCA in different fields would be a valuable contribution.

Appendix A

See Tables A.1, A.2 and A.3.

Table A.1
Unimodal benchmark functions.

Function	Dim	Range	Shift position	$f_{\min}$
$f_1(x) = \sum_{i=1}^{n} x_i^2$	20	[−100,100]	[−30, −30,..., −30]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	20	[−10,10]	[−3, −3,..., −3]	0
$f_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	20	[−100,100]	[−30, −30,..., −30]	0
$f_4(x) = \max_i\{\lvert x_i\rvert,\ 1 \le i \le n\}$	20	[−100,100]	[−30, −30,..., −30]	0
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	20	[−30,30]	[−15, −15,..., −15]	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	20	[−100,100]	[−750,..., −750]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$	20	[−1.28,1.28]	[−0.25,..., −0.25]	0

Table A.2
Multimodal benchmark functions.

Function	Dim	Range	Shift position	f_{min}
$F_8(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	20	[−500,500]	[−300,.., −300]	-418.9829×5
$F_9(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	20	[−5.12,5.12]	[−2, −2,.., −2]	0
$F_{10}(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	20	[−32,32]		0
$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	20	[−600,600]	[−400,.., −400]	0
$F_{12}(x)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$	20			
$y_i=1+\frac{x_i+1}{4}$		[−50,50]	[−30, −30,.., −30]	
$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & x_i>a\\ 0 & -a<x_i<a\\ k(-x_i-a)^m & x_i<-a\end{cases}$	20			0
$F_{13}(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+(x_n-1)^2[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$		[−50,50]	[−100,.., −100]	0

Table A.3
Composite benchmark functions.

Function	Dim	Range	f_{min}
F_{14} (CF1):			
$f_1, f_2, f_3, \ldots, f_{10}$ = Sphere Function			
$[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, .., 1]$	10	[−5,5]	0
$[\lambda_1, \lambda_2, \lambda_3 \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, .., 5/100]$			
F_{15} (CF2):			
$f_1, f_2, f_3, \ldots, f_{10}$ = *Griewank'sFunction*			
$[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, .., 1]$	10	[−5,5]	0
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/100, 5/100, 5/100, .., 5/100]$			
F_{16} (CF3):			
$f_1, f_2, f_3, \ldots, f_{10}$ = *Griewank'sFunction*			
$[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, .., 1]$	10	[−5,5]	0
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [1, 1, 1, .., 1]$			
f_{17} (CF4):			
f_1, f_2 = Ackley'sFunction			
f_3, f_4 = Rastrigin's Function			
f_5, f_6 = Weierstrass Function	10	[−5,5]	0
f_7, f_8 = Griewank's Function			
f_9, f_{10} = Sphere Function			
$[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, .., 1]$			
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [5/32,\ 5/32,\ 1,\ 1,\ 5/0.5,\ 5/0.5,\ 5/100,\ 5/100, 5/100,\ 5/100]$			
f_{18} (CF5):			
f_1, f_2 = Rastrigin's Function			
f_3, f_4 = Weierstrass Function			
f_5, f_6 = Griewank's Function	10	[−5,5]	0
f_7, f_8 = Ackley'sFunction			
f_9, f_{10} = Sphere Function			
$[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [1, 1, 1, .., 1]$			
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [1/5,\ 1/5,\ 5/0.5,\ 5/0.5,\ 5/100,\ 5/100,\ 5/32,\ 5/32, 5/100,\ 5/100]$			
f_{19} (CF6):			
f_1, f_2 = Rastrigin's Function			
f_3, f_4 = Weierstrass Function			
f_5, f_6 = Griewank's Function	10	[−5,5]	0
f_7, f_8 = Ackley'sFunction			
f_9, f_{10} = Sphere Function			
$[\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{10}] = [0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1]$			
$[\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_{10}] = [0.1*1/5,\ 0.2*1/5,\ 0.3*5/0.5,\ 0.4*5/0.5,\ 0.5*5/100,$			
$0.6*5/100, 0.7*5/32,\ 0.8*5/32, 0.9*5/100, 1*5/100]$			

Reference

[1] A.R. Simpson, G.C. Dandy, L.J. Murphy, Genetic algorithms compared to other techniques for pipe optimization, J. Water Resour. Plan. Manag. 120 (1994) 423–443.

[2] C. James, Introduction to Stochastics Search and Optimization, Wiley-Interscience, New Jersey, 2003.

[3] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inf. Sci. 237 (2013) 82–117.

[4] J.A. Parejo, A. Ruiz-Cortés, S. Lozano, P. Fernandez, Metaheuristic optimization frameworks: a survey and benchmarking, Soft Comput. 16 (2012) 527–561.

[5] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: a survey of the state of the art, Swarm Evol. Comput. 1 (2011) 32–49.

[6] S. Droste, T. Jansen, I. Wegener, Upper and lower bounds for randomized search heuristics in black-box optimization, Theory of Comput. Syst. 39 (2006) 525–544.

[7] H.H. Hoos, T. Stützle, Stochastic Local Search: Foundations & Applications, Elsevier, 2004.

[8] R.S. Parpinelli, H.S. Lopes, New inspirations in swarm intelligence: a survey, Int. J. Bio-Inspired Comput. 3 (2011) 1–16.

[9] C.M. Fonseca, P.J. Fleming, An overview of evolutionary algorithms in multiobjective optimization, Evol. Comput. 3 (1995) 1–16.

[10] A. Biswas, K. Mishra, S. Tiwari, A. Misra, Physics-inspired optimization algorithms: a survey, J. Optim. 2013 (2013).
[11] A. Gogna, A. Tayal, Metaheuristics: review and application, J. Exp. Theor. Artif. Intell. 25 (2013) 503–526.
[12] X.-S. Yang, Z. Cui, R. Xiao, A.H. Gandomi, M. Karamanoglu, Swarm intelligence and bio-inspired computation: theory and applications, Newnes (2013).
[13] S. Saremi, S. Mirjalili, A. Lewis, Biogeography-based optimisation with chaos, Neural Comput. Appl. 25 (2014) 1077–1097.
[14] G.-G. Wang, L. Guo, A.H. Gandomi, G.-S. Hao, H. Wang, Chaotic krill herd algorithm, Inf. Sci. 274 (2014) 17–34.
[15] G.-G. Wang, A. Hossein Gandomi, A. Hossein Alavi, A chaotic particle-swarm krill herd algorithm for global numerical optimization, Kybernetes 42 (2013) 962–978.
[16] G.G. Wang, S. Deb, A.H. Gandomi, Z. Zhang, A.H. Alavi, A novel cuckoo search with chaos theory and elitism scheme, in: Proceedings of 2014 International Conference on Soft Computing and Machine Intelligence (ISCMI), 2014, pp. 64–69.
[17] G.-G. Wang, S. Deb, A.H. Gandomi, Z. Zhang, A.H. Alavi, Chaotic cuckoo search, Soft Comput. (1726) 1–14.
[18] G. Wang, L. Guo, H. Wang, H. Duan, L. Liu, J. Li, Incorporating mutation scheme into krill herd algorithm for global numerical optimization, Neural Comput. Appl. 24 (2014) 853–871.
[19] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, A bat algorithm with mutation for UCAV path planning, Sci. World J. 2012 (2012).
[20] J.W. Zhang, G.G. Wang, Image matching using a bat algorithm with mutation, Appl. Mech. Mater. 203 (2012) 88–93.
[21] H.-R. Li, Y.-L. Gao, Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation, in: *Proceedings of* Second International Conference on *Information and Computing Science, 2009 (ICIC'09)*, 2009, pp. 66–69.
[22] S. Chen, Particle swarm optimization with pbest crossover, in: *Proceedings of* 2012 IEEE Congress on Evolutionary Computation (CEC), 2012, pp. 1–6.
[23] Q. Zhu, Z. Yang, An ant colony optimization algorithm based on mutation and dynamic pheromone updating, J. Softw. 15 (2004) 185–192.
[24] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer with local search, in: *Proceedings of* 2005 IEEE Congress on Evolutionary Computation, 2005, pp. 522–528.
[25] K. Premalatha, A. Natarajan, A new approach for data clustering based on PSO with local search, Comput. Inf. Sci. 1 (2008) 139.
[26] N. Noman, H. Iba, Accelerating differential evolution using an adaptive local search, IEEE Trans. Evol. Comput. 12 (2008) 107–125.
[27] J. Levine, F. Ducatelle, Ant colony optimization and local search for bin packing and cutting stock problems, J. Oper. Res. Soc. 55 (2004) 705–716.
[28] C. Blum, A. Roli, Hybrid metaheuristics: an introduction, Hybrid Metaheuristics, Springer, 2008, pp. 1–30.
[29] M. Ehrgott, X. Gandibleux, Hybrid metaheuristics for multi-objective combinatorial optimization, Hybrid metaheuristics, Springer, 2008, pp. 221–259.
[30] G. Wang, L. Guo, A novel hybrid bat algorithm with harmony search for global numerical optimization, J. Appl. Math. 2013 (2013).
[31] G.-G. Wang, A.H. Gandomi, A.H. Alavi, G.-S. Hao, Hybrid krill herd algorithm with differential evolution for global numerical optimization, Neural Comput. Appl. 25 (2014) 297–308.
[32] G. Wang, L. Guo, H. Duan, H. Wang, L. Liu, M. Shao, A hybrid metaheuristic DE/CS algorithm for UCAV three-dimension path planning, Sci. World J. 2012 (2012).
[33] G.-g. Wang, L. Guo, H. Duan, H. Wang, L. Liu, M. Shao, Hybridizing harmony search with biogeography based optimization for global numerical optimization, J. Comput. Theor. Nanosci. 10 (2013) 2312–2322.
[34] H. Duan, W. Zhao, G. Wang, X. Feng, Test-sheet composition using analytic hierarchy process and hybrid metaheuristic algorithm TS/BBO, Math. Probl. Eng. 2012 (2012).
[35] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, B. Wang, A hybrid meta-heuristic DE/CS algorithm for UCAV path planning, J. Inf. Comput. Sci. 5 (2012) 4811–4818.
[36] X. Shi, Y. Liang, H. Lee, C. Lu, L. Wang, An improved GA and a novel PSO-GA-based hybrid algorithm, Inf. Process. Lett. 93 (2005) 255–261.
[37] N. Holden, A.A. Freitas, A hybrid PSO/ACO algorithm for discovering classification rules in data mining, J. Artif. Evol. Appl. 2008 (2008) 2.
[38] S. Nemati, M.E. Basiri, N. Ghasem-Aghaee, M.H. Aghdam, A novel ACO–GA hybrid algorithm for feature selection in protein function prediction, Expert Syst. Appl. 36 (2009) 12086–12094.
[39] W.-Y. Lin, A GA–DE hybrid evolutionary algorithm for path synthesis of four-bar linkage, Mech. Mach. Theory 45 (2010) 1096–1107.
[40] B. Niu, L. Li, A novel PSO-DE-based hybrid algorithm for global optimization, Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, Springer, 2008, pp. 156–163.
[41] H. Duan, Y. Yu, X. Zhang, S. Shao, Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm, Simul. Model. Pract. Theory 18 (2010) 1104–1115.
[42] G.-G. Wang, A.H. Gandomi, X.-S. Yang, A.H. Alavi, A new hybrid method based on krill herd and cuckoo search for global optimization tasks, Int. J. Bio-Inspired Comput. (2013).
[43] G.-G. Wang, A.H. Gandomi, A.H. Alavi, An effective krill herd algorithm with migration operator in biogeography-based optimization, Appl. Math. Model. 38 (2014) 2454–2462.
[44] J.H. Holland, J.S. Reitman, Cognitive systems based on adaptive algorithms, ACM SIGART Bull. (63) (1977) 49–49.
[45] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[46] Y. Wang, H.-X. Li, T. Huang, L. Li, Differential evolution based on covariance matrix learning and bimodal distribution parameter setting, Appl. Soft Comput. 18 (2014) 232–247.
[47] Y. Wang, Z. Cai, Q. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (2011) 55–66.
[48] Y. Wang, Z. Cai, Q. Zhang, Enhancing the search ability of differential evolution through orthogonal crossover, Inf. Sci. 185 (2012) 153–177.
[49] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (2008) 702–713.
[50] I. Rechenberg, Evolutionsstrategien, in: B. Schneider, U. Ranft (Eds.), Simulationsmethoden in der Medizin und Biologie, 8, Springer, Berlin Heidelberg, 1978, pp. 83–114.
[51] M. Dorigo, M. Birattari, Ant colony optimization, Encyclopedia of Machine Learning, Springer, 2010, pp. 36–39.
[52] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.
[53] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (2007) 459–471.
[54] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (2009) 2232–2248.
[55] A. Kaveh, V. Mahdavi, Colliding Bodies Optimization method for optimum discrete design of truss structures, Comput. Struct. 139 (2014) 43–53.
[56] A. Hatamlou, Black hole: a new heuristic optimization approach for data clustering, Inf. Sci. 222 (2013) 175–184.
[57] A.H. Kashan, League Championship Algorithm (LCA): an algorithm for global optimization inspired by sport championships, Appl. Soft Comput. 16 (2014) 171–200.
[58] A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems, Appl. Soft Comput. 13 (2013) 2592–2612.
[59] R.V. Rao, V.J. Savsani, D. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput.-Aided Des. 43 (2011) 303–315.
[60] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1997) 67–82.
[61] S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, Knowl.-Based Syst. 89 (2015) 228–249.
[62] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Comput. Surv. 45 (2013) 35.
[63] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (1999) 82–102.
[64] J. Digalakis, K. Margaritis, On benchmarking functions for genetic algorithms, Int. J. Comput. Math. 77 (2001) 481–506.
[65] M. Molga, C. Smutnicki, Test functions for optimization needs, Test Funct. Optim. Needs (2005).
[66] X.-S. Yang, Test problems in optimization, 2010. Available from arXiv:1008.0549.
[67] X.-S. Yang, Firefly algorithm, stochastic test functions and design optimisation, Int. J. Bio-Inspired Comput. 2 (2010) 78–84.
[68] X.-S. Yang, A new metaheuristic bat-inspired algorithm, Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, 2010, pp. 65–74.
[69] X.-S. Yang, M. Karamanoglu, X. He, Flower pollination algorithm: a novel approach for multiobjective optimization, Eng. Optim. 46 (2014) 1222–1237.
[70] Y. Wang, Z. Cai, Combining multiobjective optimization with differential evolution to solve constrained optimization problems, IEEE Trans. Evol. Comput. 16 (2012) 117–134.
[71] S.H.R. Pasandideh, S.T.A. Niaki, A. Gharaei, Optimization of a multiproduct economic production quantity problem with stochastic constraints using sequential quadratic programming, Knowl.-Based Syst. 84 (2015) 98–107.
[72] S. Jalali, M. Seifbarghy, J. Sadeghi, S. Ahmadi, Optimizing a bi-objective reliable facility location problem with adapted stochastic measures using tuned-parameter multi-objective algorithms, Knowl.-Based Syst. (2015) in press.
[73] H. Salimi, Stochastic fractal search: a powerful metaheuristic algorithm, Knowl.-Based Syst. 75 (2015) 1–18.
[74] C.A. Coello Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Comput. Methods Appl. Mech. Eng. 191 (2002) 1245–1287.
[75] M. Drela, XFOIL: An analysis and design system for low Reynolds number airfoils, in: Low Reynolds number aerodynamics, Springer, 1989, pp. 1–12.