VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

THIS IS AN ELECTRONIC REPRINT OF THE ORIGINAL ARTICLE

**Please cite the original article:**

Necira, A., Naimi, D., Salhi, A., Salhi, S. & Menani, S. 2021. Dynamic crow search algorithm based on adaptive parameters for large-scale global optimization. Evolutionary Intelligence.

https://doi.org/10.1007/s12065-021-00628-4

**Version:** Final draft

**Copyright:** © 2021, The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature

# Dynamic crow search algorithm based on adaptive parameters for large-scale global optimization

**Abdelouahab Necira[1] · Djemai Naimi[1] · Ahmed Salhi[1] · Souhail Salhi[1] · Smail Menani[2]**

Received: 20 August 2020 / Revised: 18 April 2021 / Accepted: 29 May 2021
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

**Abstract**
Despite the good performance of Crow Search Algorithm (CSA) in dealing with global optimization problems, unfortunately it is not the case with respect to the convergence performance. Conventional CSA exploration and exploitation are strongly dependent on the proper setting of awareness probability (AP) and flight length (FL) parameters. In each optimization problem, AP and FL parameters are set in an ad hoc manner and their values do not change over the optimization process. To this date, there is no analytical approach to adjust their best values. This presents a major drawback to apply CSA in complex practical problems. Hence, the conventional CSA is used only for limited problems due to fact that CSA with fixed AP and FL is frequently trapped into local optimum. In this present paper, an enhanced version of CSA called dynamic crow search algorithm (DCSA) is proposed to overcome the drawbacks of the conventional CSA. In the proposed DCSA, two modifications of the basic algorithm are made. The first modification concerns the continuous adjustment of the CSA parameters leading to a DCSA, where AP will be adjusting linearly over optimization process and FL will be adjusting according to the generalized Pareto probability density function. This dynamic adjustment will provide more global search capability as well as more exploitation of the pre-final solutions. The second modification concerns the improvement of CSA's swarm diversity in the search process. This will lead to a high convergence accuracy, and fast convergence rate. The effectiveness of the proposed algorithm is validated using a set of experimental series using 13 complex benchmark functions. Experimental results highly proved the modified algorithm effectiveness compared to the basic algorithm in terms of convergence rate, global search capability and final solutions. In addition, a comparison with conventional and recent similar algorithms revealed that DCSA gives superior results in terms of performance and efficiency.

**Keywords** Dynamic crow search algorithm · Large scale optimization · Dynamic parameters adjustment · Benchmark functions

## 1 Introduction

Optimization problems have been widely existed in several engineering applications and science research [1]. The process of optimization is a selection of an optimal control vector in the objective function that can produce an optimal solution [2, 3]. Therefore, this solution will offer a better optimal physical process, as well as, lower cost, lower time consumption, and better performance. In the first time, several methods known as conventional optimization methods (COMs) have been proposed to solve these problems. Among these are Newton's method, gradient descent/ascend, scale conjugate gradient, and Nelder-Mead [4, 5]. COMs provide excellent performance, less time consuming, and can be easily implemented. Unfortunately, COMs have many limitations to cope with realistic and rather complicated empirical optimization problems. These limitations make COMs inefficient and often can be trapped in to local optimums, which is the major drawback of methods based on gradient [6, 7]. Hence, there is a need of developing new optimization techniques [2, 8]. Furthermore, the advance of artificial intelligence and computer science had an impact on stochastic algorithms and made them more reliable to be applied to complex real-world optimization tasks [3, 9]. These approaches initialize the optimization process with a

✉ Abdelouahab Necira
a.necira@univ-biskra.dz

[1] LGEB Laboratory, Electrical Engineering Department, Mohamed Khider University, 07000 Biskra, Algeria

[2] Information Technology Department, Vaasa University of Applied Sciences, Vaasa, Finland

set of random candidate solutions for a given problem and improve them over a pre–defined number of steps [5], and do not usually require the gradient information of problems, and they are not even sensitive for the selection of initial generation. Recently, more and more biologically-inspired approaches have been proposed and widely applied to practical problems [10].

Nature-inspired stochastic optimization (NISO) algorithms (Both as model and as metaphor) gain wide attention from the research community for decades. These algorithms either mimic individual or social behavior of a group of animal or natural phenomena, such as biological processes (e.g., reproduction, mutation, and interaction), or take advantages from species which have had adapted their physical attitude, structure, and learning to fit the environment over millions of years [3, 5, 11, 12].

The NISO algorithms start the optimization process by creating a set of random solutions that correspond to the number of problem variables. The metaheusteric algorithm is then applied to the initial solutions. Over a few iterations an improved set of solution is then generated. The fitness of all individuals (solutions) is evaluated at each iteration through the fitness function. In parallel, the resulting solutions are compared with those of previous iteration to select which solutions will be preserved to the next generation depending on algorithm strategy [10, 12]. For further information, the search process of meta-heuristic methods (MHM) is divided into two stages, exploration and exploitation [13, 14]. Firstly, exploration's aim is to lead individuals in the nearest region of the global solution, on other side, exploitation has to confine its search space into that region called promising area found previously to improve the solution. However, both of these stages are a trade-off and make the purpose of the MHM algorithms to balance between them to avoid getting trapped in local optima [4]. The MHM techniques have gained wide application in different field of science and engineering because their concept is simple, easy to implement, they proved to have quick convergence, great computational efficiency and are able to solve complicated global optimization problems.

Meta-heuristic algorithms are divided into three categories[15]. The first category is known as evolutionary algorithms. This category includes genetic algorithm (GA) [16],differential evolution (DE) [17], imperialist competitive algorithm (ICA) [18],cuckoo optimization algorithm (COA) [19]. The second category includes non-bio heuristic algorithms, such as: harmony search algorithm (HS) [20], gravitational search algorithm (GSA) [21], ions motion algorithm (IMO) [22], sine cosine algorithm (SCA) [23], exchange market algorithm (EMA) [24], teaching-learning-based optimization (TLBO) [25], mine blast algorithm (MBA) [26], soccer league competition algorithm (SLC) [27], multi-verse optimizer (MVO) [28].

The third category includes bio-inspired swarm intelligence algorithms such as: particle swarm optimization (PSO) [29], crow search algorithm (CSA) [30], butterfly optimization algorithm (BOA)[31], salp swarm algorithm (SSA)[32], grey wolf optimizer (GWO) [33], whale optimization algorithm (WOA) [34], ant lion optimizer (ALO) [35], ant colony optimization (ACO) [36], artificial bee colony (ABC) [37], bat algorithm (BA) [38], dragonfly algorithm (DA) [39], grasshopper optimization algorithm (GOA) [40], moth-flame optimization algorithm (MFO) [41], chicken swarm optimization (CSO) [42].

Askarzadeh has recently proposed a novel meta-heuristic optimizer, called crow search algorithm (CSA) [30]. CSA is a population-based optimization algorithm. CSA performs based on the idea that crows save their unneeded food in concealing places and retrieve it when it becomes in state of shortage. Therefore, crows turn into researcher in their environment for the best food source hidden by one of them, CSA algorithm has a simple mechanism controlled only by two parameters, that are: awareness probability (AP) and flight length ($fl$), a reason why it is easier to be implemented and makes it very suitable for solving complex optimization problems. In addition, CSA is able to provide optimal or near-optimal solutions for large scale optimization problems. Attracted by its simplicity and performance, researchers in many fields have applied CSA for solving complex engineering optimization problems such as: machine placement strategy in cloud data centers [43], optimal resonance-free third-order high-pass filters [44], image segmentation process[45], optimal selection conductor size in radial distribution network [46], optimal placement of STATCOM [47].

Based on "no free lunch" theorem [48], all optimization algorithms have shortcoming in solving some problems, clarifying the point that for a given algorithm, it could be well appropriate for solving a problem and provides good solutions and not for another. Therefore, there is no way to know the fittest algorithm for such problem that could be able to reach the global optimal solution in a competitive time. As other population-based optimization algorithms, CSA suffers of weak performance in some cases such as: premature convergence due to a weakness in its capacity to explore which leads to a local optimum and low convergence rate. On the other hand, conventional CSA exploration and exploitation are strongly dependent to the proper setting of AP and fl. parameters. The values of these two parameters are fixed over the process of optimization, and there is not an analytical approach to adjust their best values. The missing approach to adjust the best parameters to each problem makes CSA search process limited and it could be frequently trapped into local optimum. The main motivation of this work is to propose a solution to overcome CSA weaknesses and combine them with its powerful aspects cited previously

to extract an algorithm capable of solving a wide range of complex engineering optimization problems.

In this present paper an enhanced version of CSA called dynamic crow search algorithm (DCSA) is proposed to overcome the drawbacks of the conventional CSA. In the proposed DCSA, two modifications of the conventional algorithm are made. The first modification concerns the continuous adjustment of the CSA parameters leading to a DCSA. This will provide more global search capability as well as more exploitation of the pre-final solutions. The second modification concerns the improvement of CSA's swarm diversity in the search process. This will lead to high convergence accuracy, and fast convergence rate. The effectiveness of the proposed algorithm is verified through a set of experimental series using 13 complex benchmark functions. The experimental results compared with those of other similar algorithms revealed that DCSA can give superior results in terms of performance and efficiency.

## 2 Background

### 2.1 Overview of crow search algorithm

CSA is recent population-based optimization method, developed in 2016 by 'Alireza Askarzadeh', as its name indicates this algorithm inspiration came from an intelligent behavior of crows. Crows are considered among most intelligent animals in the world if aren't the smartest according to [49], that's owing to the fact that crow's brain to its body ration is almost or bit less than that of humans, and a lot of others genius behaviors like, they can memorize faces, they are self-aware in the mirror test, they are so skilled in using tools depending on situations and conditions, they can solve puzzles, they communicate in a sophisticated way and warn each other in case of danger, and they always score very highly on intelligence tests.

Additionally, the main inspiration of CSA is a cleverness behavior that kept the interest of 'Alireza Askarzadeh' to develop this algorithm, in a flock of crows, each crow hide its extra-food in a safety place depending on its own experience, and it come back to retrieve it when finding a new food source becomes a difficult task, and it can remember the place where the food is hidden after months. Moreover, having a relief food source don't prevent crows to search for another better food sources solicited by their greedy instinct, hence, crows follow each other in order to steal, when it comes that a crow visits its hidden place to put or retrieve food.

Stealing another's food promotes the experience of crows and protects them of being future victims, that's by taking additional precautions like moving their hidden places and predicting pilferer's behavior. From an optimization point of view, this behavior simulates an optimization process, where, crows are researchers, surrounding territory is search space, each position of the territory presents a possible solution, and quality of food source presents the objective function, as well as, the best food source is the global solution of the problem. Finally, fundamentals of CSA is relaying on those four points.

- Crows live in flock structure.
- Crows keep in mind their hidden places location.
- Crows follow each other to steal food.
- Crows get experience over time, so they protect their caches from others by probability.

#### 2.1.1 Mathematical modeling of CSA

It is assumed that the flock lives in a d-dimensional environment including a number of crows, the number of crows (population size) is mentioned by N, as well as, d corresponds to the decision variables of the problem, at each iteration crows change their positions looking for better food source, then, the position of crow i at time k (iteration) in the search space is specified by a vector, $x_{i,k} = \left[x_{i,k}^1, x_{i,k}^2, \ldots, x_{i,k}^d\right]$ for i = 1,2,….,N and k = 1,2,….,kmax, where, kmax is the maximum number of iterations. Each crow has a memory in which the position of its hiding place is memorized. At iteration k, the position of the hiding place of crow i is defined by the vector, $m_{i,k} = \left[m_{i,k}^1, m_{i,k}^2, \ldots, m_{i,k}^d\right]$, this is the best food position that crow i has obtained so far.

Assume that at iteration k, a crow j wants to check its hidden food, according to this visit we assume also that there is always another crow i following it in order to approach crow's j food, in this case, two possibilities may occur to update crow's i position.

**Case 1** crow j doesn't note that crow i is following it, therefore, crow i can reach the position of crow's j hidden place. As a result, the position of crow i is updated asfollow:

$$x_{i,k+1} = x_{i,k} + r_i \times fl \times \left(m_{j,k} - x_{i,k}\right) \tag{1}$$

Where: $r_i$ is a random number with uniform distribution between 0 and 1, while *fl* indicates the flight length, pointing out here that *fl* has great effect on search procedure, choosing *fl* lower than 1 lead the search to local solution, where the next step will be near to the $x_{i,k}$ as shown in Fig. 1a, otherwise, choosing *fl* greater than 1 lead the search to global solution because the next step will be far away than $x_{i,k}$ to promote further exploration of the search space Fig. 1b.

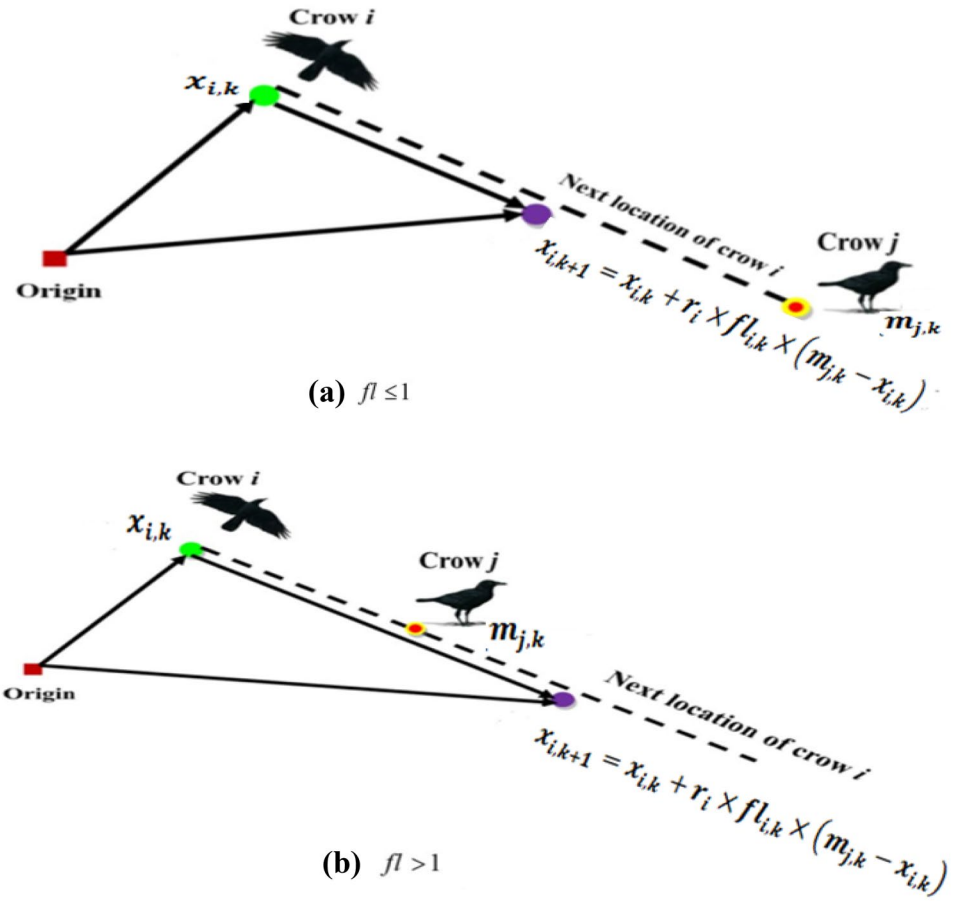


**Fig. 1** *fl* effect on the position update.

**Case 2** crow j is aware that crow i is following it, thus, crow j will dupe crow i and move randomly in the search space in order to protect its food.

Both cases can be gathered in one formula by introducing AP (awareness probability) parameter as follows:

$$x_{i,k+1} = \begin{cases} x_{i,k} + r_i \times fl \times \left(m_{j,k} - x_{i,k}\right) rj \geq AP \\ a\ random\ position\ otherwise \end{cases} \quad (2)$$

Where, $r_j$ is a random number with uniform distribution between 0 and 1, AP parameter control the algorithm intensification and diversification. Small values of AP conduct to the local search, while large values of AP conduct to global search (randomization).

The crows update their memory as follow:

$$m_{i,k+1} = \begin{cases} x_{i,k+1}\ iff\ \left(x_{i,k+1}\right) is\ better\ than\ f\left(m_{i,k}\right) \\ m_{i,k}\ otherwise \end{cases} \quad (3)$$

Where $f(.)$ symbolizes the fitness function value. Figure 2, illustrates the pseudo code of CSA.

## 2.2 Related work

Since the proposition of crow search algorithm, many researchers have applied it in multiple optimization problems, and to boost its weak side as mentioned in introduction section some modifications were proposed on literature to improve the performance of CSA.

- Modified crow search algorithm introduced in [50], proposes two modifications at the main CSA, first modification aim is to speed up the algorithm convergence, for that reason when a crow wants to generate a new position at given iteration, it must follow one of the (k) crows having best results where (k) is defined in Eq. 4, and not choosing randomly between N flock members, this selection objective is to ovoid bad solutions over iterations and can really improve the algorithm convergence and time consumption, but it can also affect the algorithm exploration in high dimension problems.

**Fig. 2** Basic CSA pseudo code

CSA algorithm

Set the algorithm initial values: N, AP, $fl$, and kmax.
Set the problem dimension
Set equal and unequal constraint
Initialize the position of each crow randomly in the search space.
Set initial memory of each crow as its first position.
Evaluate the position of each crow by the fitness function.
Set k=1 (counter initialization).
While k ≤ kmax.
For i= 1:N.
Choose randomly one member of the flock to follow for example (j).
If AP ≤ $r_j$ then.
$x_{i,k+1} = x_{i,k} + r_i \times fl \times (m_{j,k} - x_{i,k})$.
Else
$x_{i,k} = a\ random\ position\ in$ the search space.
End if.
End for.
Check the feasibility of new positions.
Update the position of each crow.
Evaluate the new positions by the fitness function.
Update the memory of each crow.
Set k= k+1.
End while.
Evaluate memory of each crow.
Extract the fittest objective function solution.
Extract the best solution.

$$K_{iter} = round\left(K_{max} - \frac{K_{max} - K_{min}}{itermax} \times iter\right) \tag{4}$$

Second modification is about adjusting the flight length parameter according to a new concept that is the distance between the crow i and its target crow j, moreover, second modification aim is to improve the algorithm exploration by choosing *fl* bigger than a threshold if the distance between the crow i and crow j is small, because in this case crow i will not improve the solution, consequently, the crow i will explore another area than the region where they are located, this modification improve the exploration but it hardly weak the exploitation, owing to the fact that in the last iterations the distance between crows get closed, and the algorithm still setting high values of *fl*, doing so the algorithm will not been focused in the promoted region, thus, it will not provide good solutions.

- Chaotic crow search algorithm introduced in [51], the main idea of the modification made by authors, is replacing the random variables of the algorithm, precisely in the formula that generate new positions, by chaotic variables came from ten different chaotic maps, subsequently, the algorithm will be formulated as follow.

$$x_{i,k+1} = \begin{cases} x_{i,k} + C_{i,k} \times fl \times \left(m_{j,k} - x_{i,k}\right) C_{j,k} \geq AP \\ a\ random\ position\ otherwise \end{cases} \tag{5}$$

Where: $C_{i,k} and C_{j,k}$ are the chaotic values resulted from the chaotic map at k iteration.

This modification improve the convergence rate and the performance of the algorithm, on the other hand, the algorithm performance is only based on making a lot of tests to have better results, and setting proper values of chaotic maps can also affect the algorithm performance.

- Rough crow search algorithm RCSA introduced in [52], authors in this article took benefits from rough set theory and integrated it with CSA to deepen the search in the promising region where the global solution is located. RCSA execution is done in two steps: firstly, CSA operates as global optimization solver to approach an approximate initial solution of a global optimization problem. Secondly, RSS (rough search scheme) is executed to ameliorate the solution quality through the roughness of the obtained optimal solution so far. Doing so, the roughness of the obtained optimal solution can be expressed as a pair of precise concepts based on the lower and upper approximations which are used to compose the interval

of boundary region. Afterward, new solutions are randomly created inside this region to enhance the diversity of solutions and achieve an effective exploration to avoid premature convergence of the swarm.

## 3 Dynamic crow search algorithm

This section is dedicated to the new concept of dynamic crow search algorithm DCSA, as mentioned previously that basic CSA suffers of premature convergence due to a weakness in its exploration capacity and low convergence rate, we could mention at this point over a deep analytic of basic CSA, that its main weakness performance came from fixed setting of its essential parameters AP and $fl$, moreover, fixed values of AP and $fl$ cannot guarantee good exploration and exploitation in the same time or they can perform well at one stage of them and not another. To fix this problem two contributions are proposed in this article, the first one affects AP parameter while the second affects $fl$ parameter to make them dynamic over iterations in favor of enhancing basic CSA performance, and extract all benefits of the search process two stages (exploration and exploitation) to have better results, detailed processes are as follow:

Firstly, awareness probability will be decreased linearly from $AP_{max}$ to $AP_{min}$ over iterations as it is shown in Fig. 3. Subjected to Eq. 6, the reason why this modification is introduced is that in the search process first stage, it is greatly recommended setting relatively high values of $AP_{max}$ to stimulate further randomization in the algorithm search process making it exploring the global search space and steer the crows to a near region where the global optimal solution is located. Afterwards, AP will get lower values until $AP_{min}$, in this stage, AP low values promote CSA crows following process Eq. 1, and eliminate almost the randomization process, doing so, DCSA will be focused on exploitation phase to extract best results from the region approached previously, while proper setting of $AP_{max}$ and $AP_{min}$ has a crucial effect on DCSA performance.

$$AP_{iter} = \frac{AP_{min} - AP_{max}}{Iter_{max}} \times iter + AP_{max} \tag{6}$$

Secondly, based in CSA Eq. 1 it can easily noticed that $fl$ parameter is multiplied by a random number with uniform distribution between 0 and 1, so that $fl$ cannot be a control parameter of CSA, because it's hardly affected by the large random variation that's multiplied by over iterations. To make $fl$ a real control parameter of CSA so that it can improve basic CSA performance the following modification is proposed where Eq. 2 will be as follow:

$$x_{i,k+1} = \begin{cases} x_{i,k} + fl_c \times \left(m_{j,k} - x_{i,k}\right) rj \geq AP \\ a\ random\ position\ otherwise \end{cases} \tag{7}$$

Where, $fl_c$ is the flight length control parameter illustrated in Fig. 4, and is defined as:

$$fl_c = \begin{cases} fl * \left[F\left(\frac{y_{max}}{10}\right) - \left(F\left(\frac{y_{max}}{10}\right) - F\left(y_{min}\right)\right) \times rand\right] if iter \leq \tau \times iter_{max}(a) \\ fl * \left[F\left(y_{max}\right) - \left(F\left(y_{max}\right) - F\left(0.6 \times y_{max}\right)\right) \times rand\right] else(b) \end{cases} \tag{8}$$

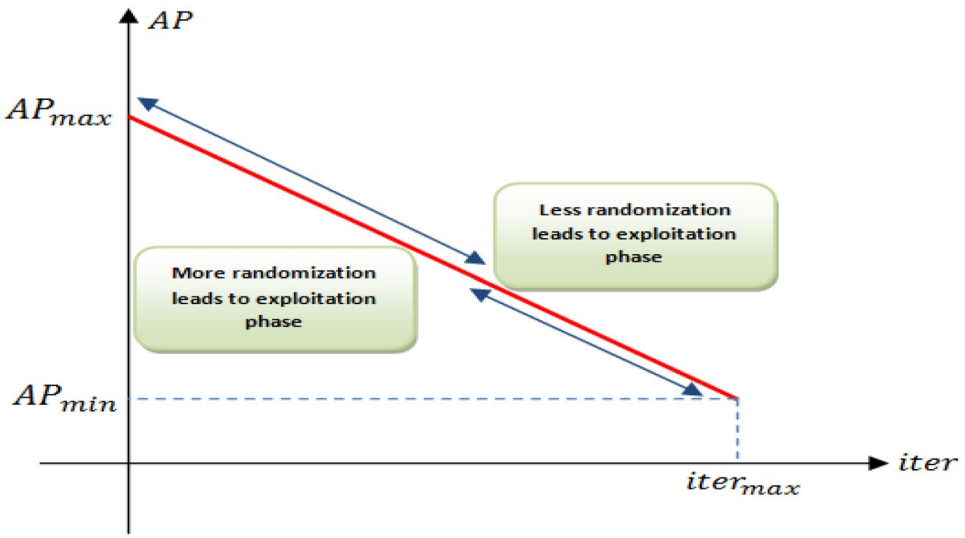


**Fig. 3** AP variation and effect on optimization search process

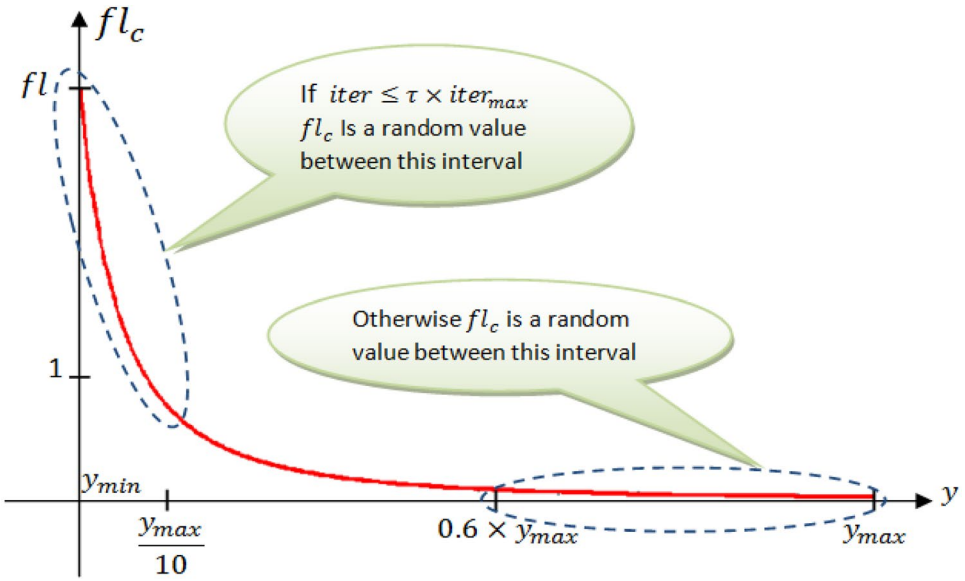


**Fig. 4** The flight length control parameter curve

Where,

$fl$: is the basic flight length.

$\tau$: is time control report, bounded between 0 and 1.

F: is the generalized Pareto probability density function, while its characteristic parameters K, SIGMA, and THETA are set 1, 1, and 0, respectively.

y: is a discontinues regular variable between $y_{min}$ and $y_{max}$, where $y_{min} = 0$ and $y_{max} = 10$, this interval is divided in 1000 uniform variables.

As it is shown in Fig. 4, the search process exploration phase is taken from the beginning of iterations to $\tau \times iter_{max}$, to provide DCSA sufficient time amount to explore the overall search space without getting trapped in local optimum, while $fl_c$ is set randomly from the region mentioned previously, this is for two reasons, firstly, to not delete the randomization process from the basic algorithm, secondly, setting $fl_c$ with relatively high values from the generalized Pareto distribution function form can control more the range variation of it, moreover, based on this variation range, $fl_c$ will be most of time between 1 and basic $fl$ so this variation range promotes more exploration than exploitation as mentioned in the background section Fig. 1. As a result, DCSA centers its search on exploration phase to reach the optimum global solution region location.

While proper setting of $\tau$ is important to guarantee a good balance between exploration and exploitation depending on the problem complexity, afterwards, in the last iterations $fl_c$ gets always lower values less than 1 from the region where Pareto distribution form takes almost steady variation from sixth-tenths $y_{max}$ to $y_{max}$, because in last iterations, almost crows get gathered in the optimum global solution region so that $fl_c$ low values boost more the exploitation on that region between the crows position that have better solutions, and don't waste main exploitation task by going to other regions where they will not improve the solution, eventually, DCSA offers the best global optimization solution, over following its own instructions showcased in DCSA flowchart Fig. 5.

## 4 Experimental results and discussion

To validate the proposed algorithm performance, DCSA is applied to thirteen selected classical benchmark functions to cover almost all perspectives that could face an optimization algorithm, while a short description of them is given below. Firstly, DCSA is compared with the basic CSA to prove its efficiency. To boost more DCSA performance some experiments are carried out to find the best parameters adjustment of it. Moreover, for the reason of abundance state of the art algorithms, DCSA is compared with four algorithms to establish its computational effectiveness over them, and so GA and PSO are selected as conventional algorithms and two other recent algorithms SSA and BOA.

### 4.1 Benchmark functions

Table 1 represents the benchmark functions used in this study and it also mentions boundaries (B), minimum value (Min), dimension (D), and type (T) of each function. Functions 1 to 7 are unimodal and they have only one optimum value, they are usually used to investigate the algorithm exploration capability and the solution convergence rate while final solution is not so important [5, 10, 53]. However, functions 8 to 13 are multimodal which are characterized by their local optimums number increasing exceptionally as the solution dimension, they are usually utilized to inspect the algorithm capability for escaping local optimums and approaching global solution,

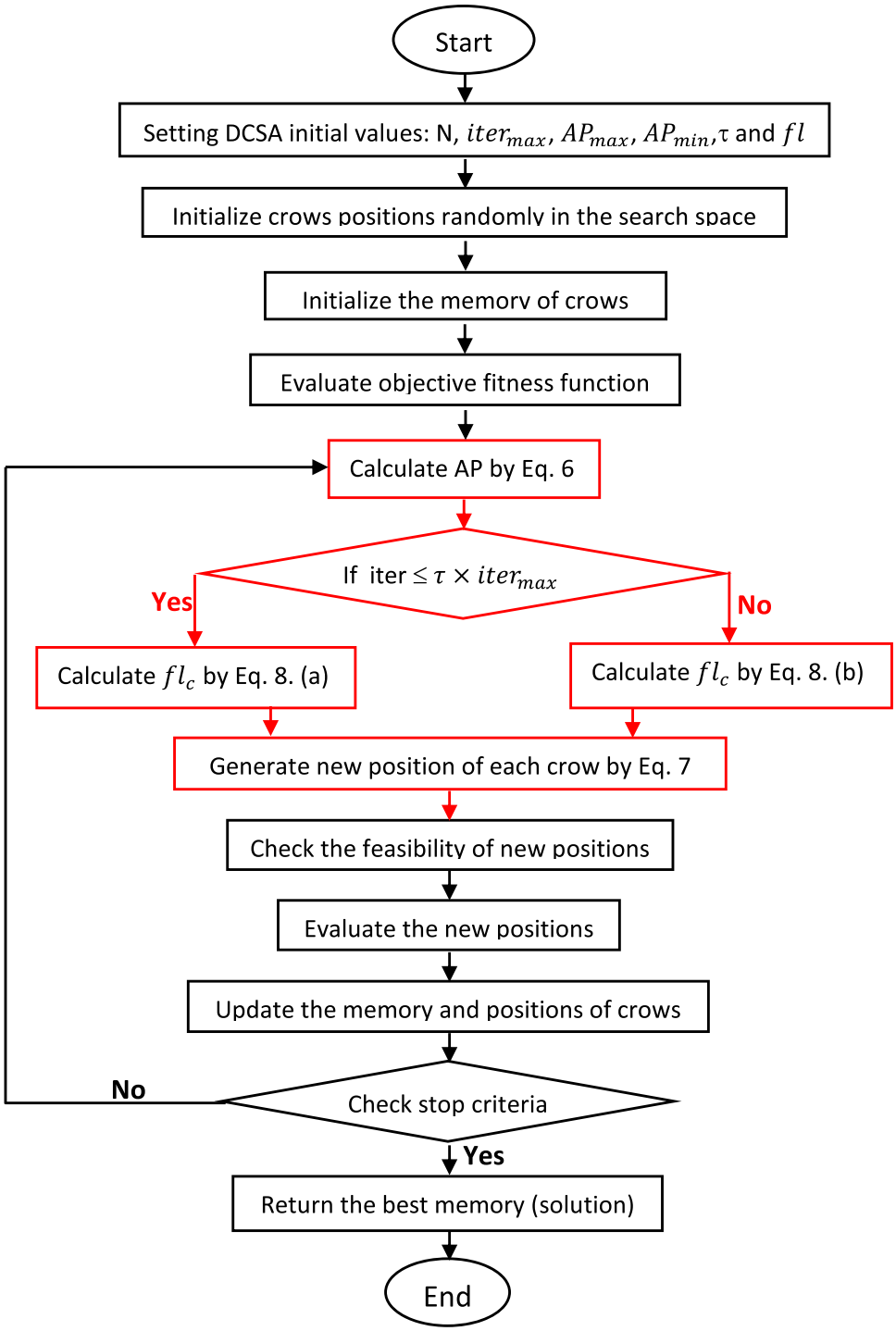


**Fig. 5** Dynamic crow search algorithm flowchart

while last solution is important to make sure that the algorithm did not get trapped in one of them.

### 4.2 Parameters settings

For all following experiments population size is set to 30, the maximum number of iteration is set to 1000, and for each function, algorithms are run 25 times independently. Performance metrics are as follow:

- Best value: is the best value that reached by an algorithm in different runs.

$$Best = Min_{1 \le i \le N_r} F_i^*(9)$$

**Table 1** Benchmark functions description

| Fun | Equation | B | Min | D | T |
|---|---|---|---|---|---|
| $f_1$ | $f(x) = \sum_{i=1}^{n} x_i^2$ | [−100, 100] | 0 | 10 | Unimodal |
| $f_2$ | $f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$ | [−10, 10] | 0 | 10 | Unimodal |
| $f_3$ | $f(x) = \sum_{i=1}^{n} \left( \sum_{j-1}^{i} x_i \right)^2$ | [−100, 100] | 0 | 10 | Unimodal |
| $f_4$ | $f(x) = max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$ | [−100, 100] | 0 | 10 | Unimodal |
| $f_5$ | $f(x) = \sum_{i=1}^{n-1} \left[ 100 \left( x_{i+1} - x_i^2 \right)^2 + \left( x_i - 1 \right)^2 \right]$ | [−30, 30] | 0 | 10 | Unimodal |
| $f_6$ | $f(x) = \sum_{i=1}^{n} \left( \left[ x_i + 0.5 \right] \right)^2$ | [−100, 100] | 0 | 10 | Unimodal |
| $f_7$ | $f(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1]$ | [-1.28, 1.28] | 0 | 10 | Unimodal |
| $f_8$ | $f(x) = \sum_{i=1}^{n} -x_i \sin \left( \sqrt{\lvert x_i \rvert} \right)$ | [−500, 500] | 418.9829×5 | 10 | Multimodal |
| $f_9$ | $f(x) = \sum_{i=1}^{n} \left[ x_i^2 - 10\cos \left( 2\pi x_i \right) + 10 \right]$ | [−5.12, 5.12] | 0 | 10 | Multimodal |
| $f_{10}$ | $f(x) = -20exp \left( -0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - exp \left( \frac{1}{n} \sum_{i=1}^{n} \cos \left( 2\pi x_i \right) \right) + 20 + e$ | [−32, 32] | 0 | 10 | Multimodal |
| $f_{11}$ | $f(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1$ | [−600, 600] | 0 | 10 | Multimodal |
| $f_{12}$ | $f(x) = \frac{\pi}{n} \left\{ 10\sin^2 \left( \pi y_1 \right) + \sum_{i=1}^{n-1} \left( y_i - 1 \right)^2 \left[ 1 + 10\sin^2 \left( \pi y_{i+1} \right) \right] + \left( y_n - 1 \right)^2 \right\} + \sum_{i=1}^{n} u \left( x_i, 10, 100, 4 \right)$ $u \left( x_i, a, k, m \right) = \begin{cases} k \left( x_i - a \right)^m, x_i > a \\ 0, -a \le x_i \le a \\ k \left( -x_i - a \right)^m, x_i < -a \end{cases}$ | [−50, 50] | 0 | 10 | Multimodal |
| $f_{13}$ | $f(x) = 0.1 \left\{ sin^2 \left( 3\pi x_1 \right) + \sum_{i=1}^{n} \left( x_i - 1 \right)^2 \left[ 1 + \sin^2 \left( 3\pi x_i + 1 \right) \right] \right.$ $\left. + \left( x_n - 1 \right)^2 \left[ 1 + \sin^2 \left( 2\pi x_n \right) \right] \right\} + \sum_{i=1}^{n} u \left( x_i, 5, 100, 4 \right)$ | [−50, 50] | 0 | 10 | Multimodal |

Where: $N_r$ is the number of different runs and $F_i^*$ is the best value.

- Average value: is the mean of the best values acquired by an algorithm of different runs.

$$Mean = \frac{1}{N_r} \sum_{i=1}^{N_r} F_i^*(10)$$

- Standard deviation: is calculated to test if an algorithm can reach the same best value in different runs and test the repeatability of the results.

$$Std = \sqrt{\frac{1}{N_r - 1} \sum_{i=1}^{N_r} (F_i^* - Mean)^2}(11)$$

For the sake of comparison, algorithms are ranked from the best to the worst according to the average value, and if two algorithms get the same average, the best value will be the second index to rank them. Table 2 represents the most recommended parameters settings of all algorithms used on following experiments extracted from their original papers, except DCSA that will be detailed section by section.

### 4.3 Effectiveness of DCSA over CSA

To point out the overcoming contribution made by DCSA above CSA, following experiment runs them independently over former benchmark functions. DCSA parameters are set to$AP_{min} = 0.01$, $AP_{max} = 0.2$, and $\tau = 0.9$, experiment results are structured in Table 3, while Fig. 6 illustrates algorithms convergence rate of some selected functions. Results from Table 1, and Fig. 6 clearly showcase that DSCA outperforms basic CSA in all unimodal functions and all multimodal functions except $f_7$ in which DCSA does not make a real improvement, this superiority is for convergence rate, quickness

**Table 2** Parameter settings of each algorithm

| algorithm | CSA | GA | PSO | BOA | SSA |
|---|---|---|---|---|---|
| parameters | AP = 0.1<br>$fl = 1.8$ | Roulette is used for selection<br>Crossover fraction = 0.9<br>Mutation = 0.005 | C1 = C2 = 2<br>$W_{max} = 0.9$<br>$W_{min} = 0.4$ | p = 0.3<br>a = 0.2<br>c = 0.02 | Without parameters |

exploration capability, avoiding local optimums, and final solutions.

| Algorithm | Index | $f_8$ | $f_9$ | $f_{10}$ | $f_{11}$ | $f_{12}$ | $f_{13}$ |
|---|---|---|---|---|---|---|---|
| CSA | Best | -2.85E+03 | 5.98E+00 | 1.66E-04 | 0.06E+00 | 1.34E-06 | 2.33E-05 |
| | Mean | −3.26E+03 | 1.31E+01 | 3.40E+00 | 0.28E+00 | 0.46E+00 | 0.12E+00 |
| | Std | 3.01E+02 | 3.96E+00 | 1.01E+00 | 0.09E+00 | 0.95E+00 | 0.21E+00 |
| | Rank | 2 | 2 | 2 | 2 | 2 | 2 |
| DCSA | Best | −2.31E+03 | 0.97E+00 | 2.83E-06 | 0.04E+00 | 4.10E-08 | 2.72E-08 |
| | Mean | −2.73E+03 | 1.05E+00 | 1.11E+00 | 0.13E+00 | 0.03E+00 | 0.01E+00 |
| | Std | 3.25E+02 | 3.14E+00 | 0.93E+00 | 0.08E+00 | 0.19E+00 | 0.02E+00 |
| | Rank | **1** | **1** | **1** | **1** | **1** | **1** |

### 4.4 Time control report and awareness probability decreasing influence

Proper setting of DCSA parameters can affect the algorithm solution process, in addition it is a tricky task to find out the best parameters adjustment for such a problem that could be different from a problem to another, this why following experiments will try to extract best parameters adjustment for each function problem. First of all, Table 4 illustrates results of running different ($AP_{max}$, $AP_{min}$) couples to each function, while $\tau$ is fixed to 0.95, and $AP_{min}$ is set always to 0.01 to guarantee a best exploitation in all cases, and $AP_{max}$ is changed to handle the exploration phase. Table 5 illustrates results of multiple time control report parameters runs at each function problem, while $AP_{min}$and $AP_{max}$ are set to 0.01 and 0.2 respectively.

Table 4 results display that various awareness probability bounds executed at each function provide different results from a couple to another this for $(f_1 f_8)$, and the best couple of previous functions is changed along them, moreover, results gap is important between different couples because some of them make a real improvement this is for $f_1$,$f_3$,$f_4 and f_5$. On the other hand, ($AP_{max} = 0.01$, $AP_{min} = 0.2$ ) is the best couple for last multimodal functions due to high value of $AP_{max}$ that promote the exploration phase on this complicated functions avoiding local optimums, so this couple is recommended for high dimensioned optimization problems.

In the other hand, $AP_{min}$ should always be set at lower values to guarantee best final solutions but avoid values very close to zero, as well as, $AP_{max}$ should not exceed 0.3 value to not simulate further randomization into search process leading to worst exploration phase. Table 5 results make clear that good setting of $\tau$ can supply a real best solutions as $f_3$,$f_4$,$f_5 and f_8$, but it depends to the function problem where carrying out some trials could be useful to find the best $\tau$ adjustment to such a problem.

### 4.5 Comparison with state of the art algorithms

The proposed algorithm should be in a high range between the computational algorithms or this improvement does not make sense, for this reason DCSA is compared with four well known alogorithms GA, PSO, BOA and SSA executed in the same benchmark problems, while DCSA parameters are the same like in (4.3) section, results are giving in Table 6 as well as the convergence rate of some selected functions in Fig. 7. Obtained results illustrate that DCSA outperforms all other algorithms in almost all functions problems, except two unimodal functions $f_5$ and $f_7$, and one multimodal function $f_9$. Superiority performance is manifested in fast and smooth convergence rate, quickness exploration capability, avoiding local optimums, and best final solutions drawn from its exploitation capability.

**Table 3** Comparison results between DCSA and CSA

| Algorithm | Index | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | $f_7$ |
|---|---|---|---|---|---|---|---|---|
| CSA | Best | 1.03E-09 | 2.38E-03 | 1.06E-03 | 2.60E-03 | 3.16E+00 | 4.59E-09 | 2.52E+00 |
| | Mean | 1.50E-07 | 0.41E+00 | 0.02E+00 | 0.01E+00 | 3.93E+01 | 2.92E-06 | 3.28E+00 |
| | Std | 1.95E-08 | 0.27E+00 | 0.01E+00 | 0.01E+00 | 2.18E+01 | 6.62E-08 | 0.40E+00 |
| | Rank | 2 | 2 | 2 | 2 | 2 | 2 | **1** |
| DCSA | Best | 4.51E-13 | 1.47E-05 | 4.95E-07 | 3.09E-05 | 0.69E+00 | 6.08E-12 | 2.32E+00 |
| | Mean | 1.37E-11 | 0.01E+00 | 5.50E-04 | 7.70E-04 | 1.26E+01 | 8.34E-10 | 3.35E+00 |
| | Std | 1.83E-09 | 0.04E+00 | 9.46E-04 | 0.00E+00 | 2.26E+00 | 9.15E-10 | 0.49E+00 |
| | Rank | **1** | **1** | **1** | **1** | **1** | **1** | 2 |

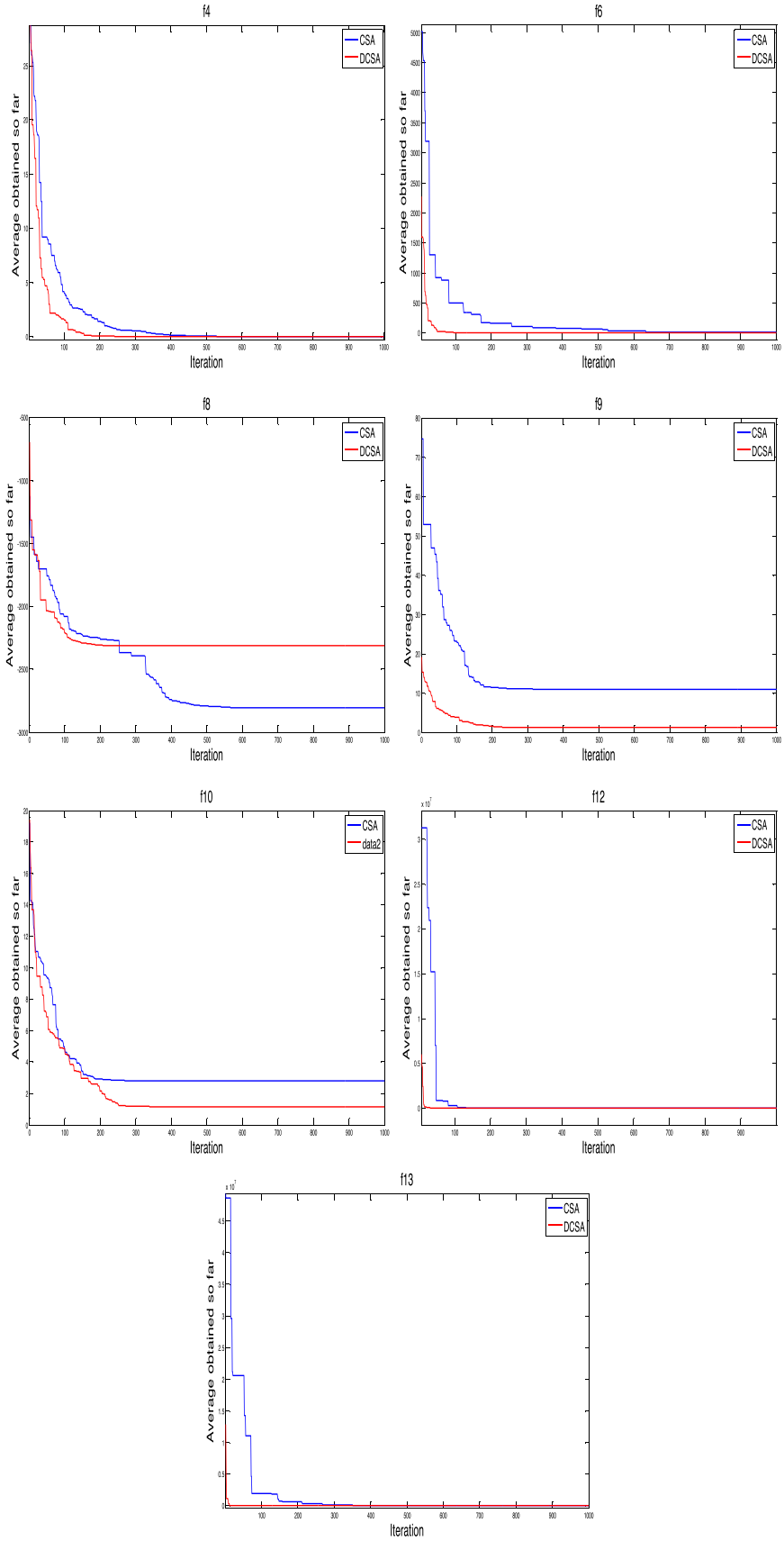


**Fig. 6** Comparative convergence rates profiles

**Table 4** Comparison results of different AP couples

| Function | Index | $AP_{min} = 0.01$ $AP_{max} = 0.2$ | $AP_{min} = 0.01$ $AP_{max} = 0.15$ | $AP_{min} =0.01$ $AP_{max} = 0.1$ | $AP_{min} =0.01$ $AP_{max} = 0.05$ |
|---|---|---|---|---|---|
| $f_1$ | Best | 2.18E-12 | 1.53E-12 | 2.11E-13 | 3.13E-13 |
| | Mean | 2.16E-10 | 6.62E-11 | 2.10E-11 | 2.07E-09 |
| | Std | 2.95E-10 | 9.96E-11 | 3.66E-11 | 3.88E-09 |
| | Rank | 3 | 2 | **1** | 4 |
| $f_2$ | Best | 3.67E-05 | 4.73E-06 | 6.10E-05 | 7.03E-05 |
| | Mean | 0.01E + 00 | 0.00E + 00 | 0.01E + 00 | 0.17E + 00 |
| | Std | 0.04E + 00 | 0.00E + 00 | 0.02E + 00 | 0.39E + 00 |
| | Rank | 2 | **1** | 3 | 3 |
| $f_3$ | Best | 1.65E-05 | 7.50E-07 | 6.83E-06 | 2.82E-04 |
| | Mean | 3.47Ee-04 | 5.65E-04 | 0.00E + 00 | 0.02E + 00 |
| | Std | 4.25E-04 | 9.59E-04 | 0.00E + 00 | 0.05E + 00 |
| | Rank | **1** | 2 | 3 | 4 |
| $f_4$ | Best | 2.49E-05 | 4.25E-05 | 7.47E-05 | 6.16E-05 |
| | Mean | 0.00E + 00 | 8.18E-04 | 0.00E + 00 | 0.01E + 00 |
| | Std | 0.00E + 00 | 8.78E-04 | 0.00E + 00 | 0.01E + 00 |
| | Rank | 2 | **1** | 3 | 3 |
| $f_5$ | Best | 2.01E + 00 | 1.79E + 00 | 1.99E + 00 | 3.07E + 00 |
| | Mean | 1.04E + 01 | 2.43E + 01 | 1.01E + 01 | 2.60E + 01 |
| | Std | 1.77E + 01 | 7.55E + 01 | 1.75E + 01 | 8.17E + 01 |
| | Rank | 2 | 3 | **1** | 4 |
| $f_6$ | Best | 3.98E-12 | 1.49E-13 | 1.79E-12 | 3.32E-12 |
| | Mean | 2.93E-10 | 6.96E-11 | 1.55E-10 | 7.45E-10 |
| | Std | 3.67E-10 | 1.55E-10 | 2.72E-10 | 1.96E-09 |
| | Rank | 3 | **1** | 2 | 4 |
| $f_7$ | Best | 2.41E + 00 | 2.61E + 00 | 1.77E + 00 | 2.24E + 00 |
| | Mean | 3.22 E + 00 | 3.38E + 00 | 3.21E + 00 | 3.02 E + 00 |
| | Std | 0.35E + 00 | 0.34E + 00 | 0.47E + 00 | 0.49E + 00 |
| | Rank | 3 | 4 | 2 | **1** |
| $f_8$ | Best | −2.35E + 03 | −2.23E + 03 | −2.38E + 03 | −2.10E + 03 |
| | Mean | −2.63E + 03 | −2.62E + 03 | −2.78E + 03 | −2.58E + 03 |
| | Std | 2.97E + 02 | 3.16E + 02 | 4.37E + 02 | 3.44E + 02 |
| | Rank | 3 | 2 | 4 | **1** |
| $f_9$ | Best | 2.98E + 00 | 4.97E + 00 | 2.98E + 00 | 2.98E + 00 |
| | Mean | 3.99E + 00 | 8.11E + 00 | 5.01E + 00 | 6.11E + 00 |
| | Std | 3.74E + 00 | 4.27E + 00 | 4.77E + 00 | 4.57E + 00 |
| | Rank | **1** | 4 | 2 | 3 |
| $f_{10}$ | Best | 8.43E-07 | 6.93E-07 | 2.33E-06 | 5.67E-06 |
| | Mean | 1.46E + 00 | 1.77E + 00 | 1.97E + 00 | 2.84E + 00 |
| | Std | 1.00E + 00 | 0.80E + 00 | 0.88E + 00 | 0.97E + 00 |
| | Rank | **1** | 2 | 3 | 4 |
| $f_{11}$ | Best | 0.02E + 00 | 0.03E + 00 | 0.03E + 00 | 0.06E + 00 |
| | Mean | 0.13E + 00 | 0.14E + 00 | 0.17E + 00 | 0.24E + 00 |
| | Std | 0.08E + 00 | 0.08E + 00 | 0.09E + 00 | 0.17E + 00 |
| | Rank | **1** | 2 | 3 | 4 |
| $f_{12}$ | Best | 9.16E-09 | 2.73E-08 | 1.14E-08 | 6.07E-06 |
| | Mean | 0.08E + 00 | 0.12E + 00 | 0.30E + 00 | 0.98E + 00 |
| | Std | 0.13E + 00 | 0.25E + 00 | 0.43E + 00 | 1.06E + 00 |
| | Rank | **1** | 2 | 3 | 4 |
| $f_{13}$ | Best | 2.61E-07 | 2.88E-07 | 2.73E-09 | 8.74E-05 |
| | Mean | 0.01E + 00 | 0.01E + 00 | 0.03E + 00 | 0.13E + 00 |
| | Std | 0.01E + 00 | 0.01E + 00 | 0.06E + 00 | 0.18E + 00 |
| | Rank | **1** | 2 | 3 | 4 |

## 5 Conclusions

In this paper, the original CSA is improved by introducing two modifications into the main algorithm. Hence, DCSA gain more advantages like the algorithm search process becomes dynamic and promising the exploration and exploitation phases independently by mean of adaptive parameters, DCSA convergence rate is faster

**Table 5** Comparison results of different time control report values

| Function | Index | $\tau = 0.8$ | $\tau = 0.85$ | $\tau = 0.9$ | $\tau = 0.95$ |
|---|---|---|---|---|---|
| $f_1$ | Best | 2.01E-09 | 7.12E-11 | 1.42E-11 | 9.47E-12 |
| | Mean | 1.25E-08 | 5.52E-09 | 9.07E-10 | 3.46E-10 |
| | Std | 1.28E-08 | 1.14E-08 | 1.45E-09 | 7.58E-10 |
| | Rank | 4 | 3 | 2 | **1** |
| $f_2$ | Best | 1.16E-04 | 7.98E-06 | 1.90E-05 | 1.98E-05 |
| | Mean | 0.04E+00 | 0.02E+00 | 0.03E+00 | 0.13E+00 |
| | Std | 0.09E+00 | 0.09E+00 | 0.09E+00 | 0.35E+00 |
| | Rank | 3 | **1** | 2 | 4 |
| $f_3$ | Best | 2.57E-05 | 2.24E-05 | 3.29E-07 | 2.60E-05 |
| | Mean | 0.00E+00 | 0.00E+00 | 3.30E-04 | 3.77E-04 |
| | Std | 0.00E+00 | 0.00E+00 | 4.43E-04 | 5.99E-04 |
| | Rank | 4 | 3 | **1** | 2 |
| $f_4$ | Best | 1.69E-04 | 7.84E-05 | 5.07E-05 | 4.91E-05 |
| | Mean | 0.00E+00 | 0.00E+00 | 4.65E-04 | 8.77E-04 |
| | Std | 0.00E+00 | 0.00E+00 | 4.44E-04 | 0.00E+00 |
| | Rank | 4 | 3 | **1** | 2 |
| $f_5$ | Best | 0.70E+00 | 3.00E+00 | 4.90E+00 | 1.36E+00 |
| | Mean | 1.84E+01 | 2.40E+01 | 1.32E+01 | 1.21E+01 |
| | Std | 5.31E+01 | 5.36E+01 | 4.69E+01 | 3.19E+01 |
| | Rank | 3 | 4 | 2 | **1** |
| $f_6$ | Best | 3.10E-10 | 8.64E-11 | 5.26E-11 | 8.44E-12 |
| | Mean | 1.87E-08 | 3.32E-09 | 1.20E-09 | 1.83E-10 |
| | Std | 2.06E-08 | 3.85E-09 | 1.93E-09 | 2.66E-10 |
| | Rank | 4 | 3 | 2 | **1** |
| $f_7$ | Best | 2.73E+00 | 1.85E+00 | 2.13E+00 | 2.16E+00 |
| | Mean | 3.42E+00 | 3.32E+00 | 3.25E+00 | 3.34E+00 |
| | Std | 0.36E+00 | 0.49E+00 | 0.42E+00 | 0.43E+00 |
| | Rank | 4 | 2 | **1** | 3 |
| $f_8$ | Best | −2.35E+03 | −2.34E+03 | −2.45E+03 | −2.12E+03 |
| | Mean | −2.71E+03 | −2.70E+03 | −2.75E+03 | −2.69E+03 |
| | Std | 3.39E+02 | 2.95E+02 | 2.75E+02 | 3.12E+02 |
| | Rank | 3 | 2 | 4 | **1** |
| $f_9$ | Best | 3.97E+00 | 3.97E+00 | 3.97E+00 | 1.98E+00 |
| | Mean | 1.03E+01 | 1.20E+00 | 1.11E+00 | 5.94E+00 |
| | Std | 4.54E+00 | 5.54E+00 | 5.00E+00 | 3.43E+00 |
| | Rank | 4 | 2 | **1** | 3 |
| $f_{10}$ | Best | 1.07E-05 | 7.36E-06 | 4.15E-06 | 2.96E-06 |
| | Mean | 0.93E+00 | 1.05E+00 | 1.15E+00 | 1.22E+00 |
| | Std | 0.91E+00 | 0.98E+00 | 1.03E+00 | 0.92E+00 |
| | Rank | **1** | 2 | 3 | 4 |
| $f_{11}$ | Best | 0.03E+00 | 0.02E+00 | 0.03E+00 | 0.03E+00 |
| | Mean | 0.15E+00 | 0.11E+00 | 0.12E+00 | 0.14E+00 |
| | Std | 0.06E+00 | 0.06E+00 | 0.08E+00 | 0.06E+00 |
| | Rank | 4 | 1 | 2 | 3 |
| $f_{12}$ | Best | 2.04E-07 | 4.35E-08 | 1.00E-07 | 1.13E-07 |
| | Mean | 0.13E+00 | 0.18E+00 | 0.25E+00 | 0.21E+00 |
| | Std | 0.18E+00 | 0.32E+00 | 0.48E+00 | 0.49E+00 |
| | Rank | **1** | 2 | 4 | 3 |
| $f_{13}$ | Best | 2.79E-06 | 1.16E-06 | 2.72E-08 | 1.32E-07 |
| | Mean | 0.01E+00 | 0.02E+00 | 0.01E+00 | 0.01E+00 |
| | Std | 0.01E+00 | 0.03E+00 | 0.01E+00 | 0.01E+00 |
| | Rank | 3 | 4 | **1** | 2 |

toward the final solution, DCSA dynamic parameters are independent to each problem and the final solution is most of time the best one. Multiple benchmark problems was used to assess the performance of the proposed algorithm, as well as, some experiments has been carried out to extract the best parameter adjustment of the new algorithm for each benchmark problem, finally, the proposed algorithm has been compared to those surfaced in the powerful and recent optimization field. Results obtained validate the effectiveness and

**Table 6** Comparison results between DCSA and some state of the art algorithms

| Function | Index | DCSA | GA | PSO | BOA | SSA |
|---|---|---|---|---|---|---|
| $f_1$ | Best | 4.51E-13 | 1.78E-09 | 1.74E-04 | 3.23E-04 | 2.48E-10 |
| | Mean | 1.37E-11 | 5.05E-08 | 0.09E+00 | 3.56E-04 | 7.20E-10 |
| | Std | 1.83E-09 | 2.51E-08 | 0.13E+00 | 1.91E-05 | 1.88E-10 |
| | Rank | **1** | 3 | 5 | 4 | 2 |
| $f_2$ | Best | 1.47E-05 | 1.11E-03 | 0.02E+00 | 0.00E+00 | 1.11E-04 |
| | Mean | 0.10E-02 | 0.55E+00 | 1.34E+00 | 0.00E+00 | 0.00E+00 |
| | Std | 0.04E+00 | 0.69E+00 | 3.23E+00 | 1.95E-04 | 0.09E+00 |
| | Rank | **1** | 4 | 5 | 3 | 2 |
| $f_3$ | Best | 4.95E-07 | 6.57E-06 | 0.00E+00 | 2.94E-04 | 2.94E-06 |
| | Mean | 5.50E-04 | 7.35E-03 | 3.19E+00 | 3.42E-03 | 8.42E-04 |
| | Std | 9.46E-04 | 0.00E+00 | 4.43E+00 | 1.96E-05 | 1.96E-04 |
| | Rank | **1** | 4 | 5 | 3 | 2 |
| $f_4$ | Best | 3.09E-05 | 0.50E+00 | 0.09E+00 | 0.00E+00 | 4.05E-05 |
| | Mean | 7.70E-04 | 1.28E+00 | 0.74E+00 | 0.00E+00 | 6.80E-02 |
| | Std | 0.00E+00 | 0.43E+00 | 0.57E+00 | 1.77E-04 | 0.00E+00 |
| | Rank | **1** | 5 | 4 | 3 | 2 |
| $f_5$ | Best | 0.69E+00 | 2.69E+00 | 0.48E+00 | 8.89E+00 | 2.92E+00 |
| | Mean | 1.26E+01 | 1.48E+01 | 1.44E+02 | 8.94E+00 | 1.18E+02 |
| | Std | 2.26E+00 | 7.01E+00 | 3.26E+02 | 0.02E+00 | 3.04E+02 |
| | Rank | 2 | 3 | 5 | **1** | 4 |
| $f_6$ | Best | 6.08E-12 | 6.59E-10 | 3.03E-04 | 0.50E+00 | 4.02E-10 |
| | Mean | 8.34E-10 | 5.59E-08 | 0.09E+00 | 1.22E+00 | 6.35E-07 |
| | Std | 9.15E-10 | 3.06E-08 | 0.11E+00 | 0.33E+00 | 1.61E-07 |
| | Rank | **1** | 2 | 4 | 5 | 3 |
| $f_7$ | Best | 2.32E+00 | 2.36E+00 | 2.39E+00 | 1.09E+00 | 1.54E+00 |
| | Mean | 3.35E+00 | 3.51E+00 | 3.38E+00 | 1.57E+00 | 2.15E+00 |
| | Std | 0.49E+00 | 0.48E+00 | 0.41E+00 | 0.20E+00 | 0.30E+00 |
| | Rank | 3 | 5 | 4 | **1** | 2 |
| $f_8$ | Best | −2.31E+03 | −2.88E+03 | −2.76E+03 | −1.82E+03 | −2.47E+03 |
| | Mean | −2.73E+03 | −3.25E+03 | −3.22E+03 | −1.26E+03 | −2.84E+03 |
| | Std | 3.25E+02 | 3.21E+02 | 3.00E+02 | 2.93E+02 | 3.35E+02 |
| | Rank | **1** | 3 | 2 | 4 | 2 |
| $f_9$ | Best | 0.97E+00 | 1.22E-06 | 2.01E+00 | 3.85E-02 | 5.96E+00 |
| | Mean | 1.05E+00 | 5.69E+00 | 1.46E+01 | 4.14E-02 | 2.34E+01 |
| | Std | 3.14E+00 | 4.65E+00 | 1.05E+01 | 1.56E-03 | 8.40E+00 |
| | Rank | 2 | 3 | 4 | **1** | 5 |
| $f_{10}$ | Best | 2.83E-06 | 3.64E-05 | 0.00E+00 | 0.00E+00 | 8.46E-06 |
| | Mean | 1.11E+00 | 4.11E+00 | 2.64E+00 | 2.90E+00 | 3.36E+00 |
| | Std | 0.93E+00 | 0.96E+00 | 3.78E+00 | 1.41E-04 | 7.23E+00 |
| | Rank | **1** | 5 | 2 | 3 | 4 |
| $f_{11}$ | Best | 7.76E-06 | 0.04E+00 | 0.04E+00 | 8.46E-04 | 0.07E+00 |
| | Mean | 0.00E+00 | 0.13E+00 | 0.23E+00 | 0.13E+00 | 0.24E+00 |
| | Std | 0.01E+00 | 0.08E+00 | 0.12E+00 | 0.15E+00 | 0.11E+00 |
| | Rank | **1** | 3 | 4 | 2 | 5 |
| $f_{12}$ | Best | 4.10E-08 | 5.53E-09 | 3.85E-05 | 0.13E+00 | 6.27E-07 |
| | Mean | 0.00E+00 | 0.08E+00 | 0.19E+00 | 0.53E+00 | 0.15E+00 |
| | Std | 0.19E+00 | 0.20E+00 | 0.26E+00 | 0.31E+00 | 0.14E+00 |
| | Rank | **1** | 2 | 4 | 5 | 3 |
| $f_{13}$ | Best | 2.72E-11 | 6.94E-09 | 9.50E-04 | 0.40E+00 | 3.45E-09 |
| | Mean | 0.01E+00 | 0.44E+00 | 0.47E+00 | 0.88E+00 | 0.75E+00 |
| | Std | 0.02E+00 | 0.14E+00 | 0.07E+00 | 0.20E+00 | 0.00E+00 |
| | Rank | **1** | 2 | 3 | 5 | 4 |

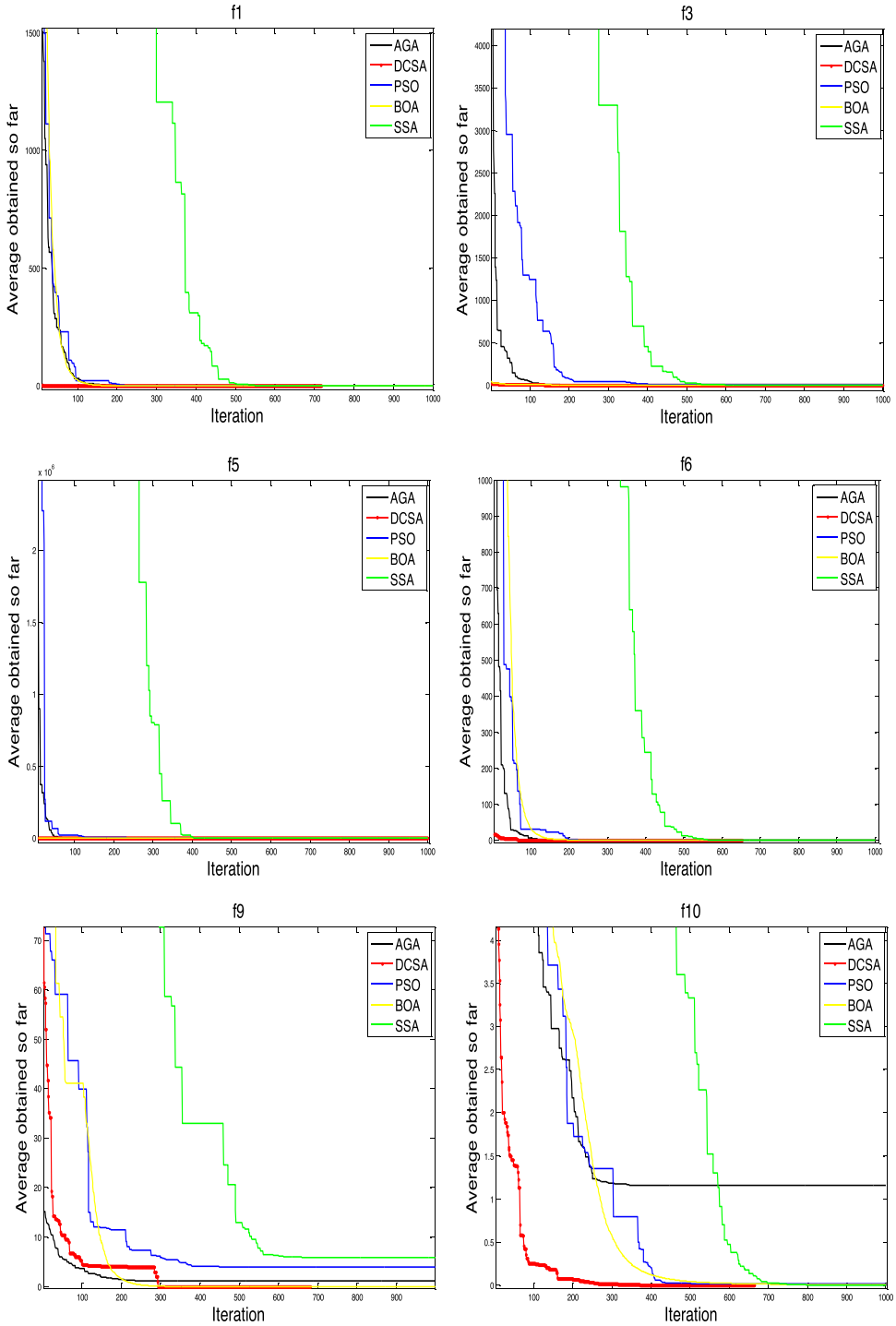


**Fig. 7** Comparative convergence rates profiles

superiority of the new algorithm that it can overtake the original algorithm weaknesses, promising its application to the practical and engineering problems for further research in which it can challenge high dimension and additional constraint problems that can effects its performance.

## References


1. Mansour IB, I. Alaya, *and* M. Tagina, "A gradual weight-based ant colony approach for solving the multiobjective multidimensional knapsack problem" Evol Intel, *vol.* 12, 253–2722019
2. Wang L, J. Pei, Wen Y, J. Pi, Fei M, *and* Pardalos PM, An improved adaptive human learning algorithm for engineering optimization Appl Soft Comput, *vol.* 71, 894–9042018
3. Chen K, F. Zhou, Wang Y, *and* Yin L, An ameliorated particle swarm optimizer for solving numerical optimization problems Appl Soft Comput, *vol.* 73, 482–4962018
4. Singh PR, M. A. Elaziz, *and* S. Xiong, "Modified Spider Monkey Optimization based on Nelder–Mead method for global optimization" Expert Syst Appl, *vol.* 110, 264–2892018
5. Ewees AA, M. Abd Elaziz, *and* E. H. Houssein, "Improved grasshopper optimization algorithm using opposition-based learning," Expert Syst Appl, *vol.* 112, 156–1722018
6. Mansour IB, Alaya I (2015) Indicator based ant colony optimization for multi-objective knapsack problem. Procedia Comput Sci *vol.* 60:448–457
7. Mansour IB, M. Basseur, *and* F. Saubion, "A multi-population algorithm for multi-objective knapsack problem" Appl Soft Comput, *vol.* 70, 814–8252018
8. Shi H, S. Liu, Wu H, R. Li, Liu S, N. Kwok, and Y. Peng, "Oscillatory Particle Swarm Optimizer" *Appl Soft Comput*, vol. 73, 316–3272018
9. Omran MGH, S. Alsharhan, *and* M. Clerc, "A modified Intellects-Masses Optimizer for solving real-world optimization problems" Swarm Evolutionary Computation, vol. 41, 159–1662018
10. Sun Y, X. Wang, Chen Y, *and* Liu Z, "A modified whale optimization algorithm for large-scale global optimization problems" Expert Syst Appl, *vol.* 114, 563–5772018
11. Shaw B, V. Mukherjee, *and* S. P. Ghoshal, "A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems" *Int J Electric Power Energy Syst*, *vol.* 35, 21–332012
12. Nenavath H, D. R. Kumar Jatoth, *and* D. S. Das, "A synergy of the sine-cosine algorithm and particle swarm optimizer for improved global optimization and object tracking" Swarm Evolut Comput, *vol.* 43, 1–302018
13. Mansour IB, I. Alaya, *and* M. Tagina, *"Chebyshev-based iterated local search for multi-objective optimization," in* 2017 *13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), 2017, pp. 163–170.*
14. Ben Mansour I, I. Alaya, *and* M. Tagina, *"A min-max Tchebycheff based local search approach for MOMKP," in Proceedings of the 12th International Conference on Software Technologies, ICSOFT, INSTICC, pp. 140–150.*
15. Torabi S, Safi-Esfahani F (2018) "Improved Raven Roosting Optimization algorithm (IRRO)". Swarm Evolut Comput *vol.* 40:144–154
16. Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press
17. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim *vol.* 11:341–359
18. Atashpaz-Gargari E, Lucas C, *"Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition," in Evolutionary computation, 2007. CEC 2007. IEEE Congress on*, 2007, *pp. 4661–4667.*
19. Rajabioun R (2011) "Cuckoo Optimization Algorithm". Appl Soft Comput *vol.* 11:5508–5518
20. Geem ZW, J. H. Kim, *and* G. V. Loganathan, *"A new heuristic optimization algorithm: harmony search," simulation, vol. 76, pp. 60–68*, 2001
21. Rashedi E, H. Nezamabadi-pour, *and* S. Saryazdi, *"GSA: A Gravitational Search Algorithm," Information Sciences, vol. 179, pp. 2232–2248*, 2009
22. Javidy B, A. Hatamlou, *and* S. Mirjalili, Ions motion algorithm for solving optimization problems Appl Soft Comput, *vol.* 32, 72–792015
23. Mirjalili S (2016) "SCA: A Sine Cosine Algorithm for solving optimization problems". Knowl-Based Syst *vol.* 96:120–133
24. Ghorbani N, Babaei E (2014) Exchange market algorithm. Appl Soft Comput *vol.* 19:177–187
25. Rao RV, V. J. Savsani, *and* D. P. Vakharia, Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems Comput Aided Des, *vol.* 43, 303–3152011
26. Sadollah A, A. Bahreininejad, Eskandar H, *and* Hamdi M, Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems Appl Soft Comput, *vol.* 13, 2592–26122013
27. Moosavian N, Roodsari BK (2014) Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks. Swarm Evolut Comput *vol.* 17:14–24
28. Mirjalili S, S. M. Mirjalili, *and* A. Hatamlou, "Multi-Verse Optimizer: a nature-inspired algorithm for global optimization" Neural Comput Appl, *vol.* 27, 495–5132015
29. Eberhart R, Kennedy J, *"A new optimizer using particle swarm theory," in Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, 1995, *pp. 39–43.*
30. Askarzadeh A (2016) "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm". *Comput Struct vol.* 169:1–12
31. Arora S, Singh S (2019) Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput *vol.* 23:715–734
32. Mirjalili S, A. H. Gandomi, Mirjalili SZ, S. Saremi, Faris H, *and* Mirjalili SM, "Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems" Adv Eng Softw, *vol.* 114, 163–1912017
33. Mirjalili S, S. M. Mirjalili, *and* A. Lewis, "Grey Wolf Optimizer" *Adv Eng Softw*, *vol.* 69, 46–612014
34. Mirjalili S, Lewis A (2016) The Whale Optimization Algorithm. Adv Eng Softw *vol.* 95:51–67
35. Mirjalili S (2015) "The Ant Lion Optimizer". Adv Eng Softw *vol.* 83:80–98
36. Dorigo M, Di Caro G, *"Ant colony optimization: a new metaheuristic," in Proceedings of the* (1999) *congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), 1999, pp. 1470–1477.*
37. Karaboga D, Basturk B (2007) "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm". J Global Optim *vol.* 39:459–471
38. Yang X-S (2010) "A new metaheuristic bat-inspired algorithm". In: *in* Nature inspired cooperative strategies for optimization (NICSO 2010). *ed*: Springer, pp 65–74
39. Mirjalili S (2015) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl *vol.* 27:1053–1073
40. Saremi S, S. Mirjalili, *and* A. Lewis, "Grasshopper Optimisation Algorithm: Theory and application" Adv Eng Softw, *vol.* 105, 30–472017
41. Mirjalili S (2015) "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm". Knowl-Based Syst *vol.* 89:228–249
42. Meng X, Y. Liu, Gao X, *and* Zhang H, *"A new bio-inspired algorithm: chicken swarm optimization," in International conference in swarm intelligence*, 2014, *pp. 86–94.*
43. Satpathy A, S. K. Addya, Turuk AK, B. Majhi, *and* G. Sahoo, Crow search based virtual machine placement strategy in cloud data centers with live migration *Comput Electric Engi*, *vol.* 69, 334–3502018
44. Aleem SHA, A. F. Zobaa, *and* M. E. Balci, Optimal resonance-free third-order high-pass filters based on minimization of the total cost

of the filters using Crow Search Algorithm Electr Power Syst Res, *vol.* 151, 381–3942017
45. Oliva D, S. Hinojosa, Cuevas E, G. Pajares, Avalos O, *and* Gálvez J, Cross entropy based thresholding for magnetic resonance brain images using Crow Search Algorithm Expert Syst Appl, *vol.* 79, 164–1802017
46. Abdelaziz AY, Fathy A (2017) A novel approach based on crow search algorithm for optimal selection of conductor size in radial distribution networks. Eng Sci Technol Int J *vol.* 20:391–402
47. Choudhary G, N. Singhal, *and* K. Sajan, *"Optimal placement of STATCOM for improving voltage profile and reducing losses using crow search algorithm," in Control, Computing, Communication and Materials (ICCCCM), 2016 International Conference on*, 2016, *pp. 1–6.*
48. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE transactions on evolutionary computation *vol.* 1:67–82
49. Gupta D, J. J. Rodrigues, Sundaram S, A. Khanna, Korotaev V, *and* de Albuquerque VHC, "Usability feature extraction using modified crow search algorithm: a novel approach" *Neural Computi Appl*, *pp.* 1–11, 2018
50. Mohammadi F, Abdi H (2018) "A modified crow search algorithm (MCSA) for solving economic load dispatch problem". Appl Soft Comput *vol.* 71:51–65
51. Sayed GI, A. E. Hassanien, *and* A. T. Azar, Feature selection via a novel chaotic crow search algorithm Neural Comput Appl, *vol.* 31, 171–1882019
52. Hassanien AE, R. M. Rizk-Allah, *and* M. Elhoseny, "A hybrid crow search algorithm based on rough searching scheme for solving engineering optimization problems" *J Ambient Intelli Human Comput*, *pp.* 1–25, 2018
53. Luo J, Shi B (2019) A hybrid whale optimization algorithm based on modified differential evolution for global optimization problems. Appl Intell *vol.* 49:1982–2000

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.