scientific reports

OPEN

The cheetah optimizer: a nature-inspired metaheuristic algorithm for large-scale optimization problems

Mohammad Amin Akbari[1], Mohsen Zare[2], Rasoul Azizipanah-abarghooee[3], Seyedali Mirjalili[4,5] & Mohamed Deriche[1]✉

Motivated by the hunting strategies of cheetahs, this paper proposes a nature-inspired algorithm called the cheetah optimizer (CO). Cheetahs generally utilize three main strategies for hunting prey, i.e., searching, sitting-and-waiting, and attacking. These strategies are adopted in this work. Additionally, the leave the pray and go back home strategy is also incorporated in the hunting process to improve the proposed framework's population diversification, convergence performance, and robustness. We perform intensive testing over 14 shifted-rotated CEC-2005 benchmark functions to evaluate the performance of the proposed CO in comparison to state-of-the-art algorithms. Moreover, to test the power of the proposed CO algorithm over large-scale optimization problems, the CEC2010 and the CEC2013 benchmarks are considered. The proposed algorithm is also tested in solving one of the well-known and complex engineering problems, i.e., the economic load dispatch problem. For all considered problems, the results are shown to outperform those obtained using other conventional and improved algorithms. The simulation results demonstrate that the CO algorithm can successfully solve large-scale and challenging optimization problems and offers a significant advantage over different standards and improved and hybrid existing algorithms. Note that the source code of the CO algorithm is publicly available at https://www.optim-app.com/projects/co.

Background. Recently, solving optimization problems have become a challenging and exciting topic in most research area. Decision-making problems that are overgrowing can be defined as optimization problems. An optimization problem includes one or more objective functions, decision variables, and constraints to be minimized or maximized[1]. Many deterministic approaches, such as linear programming, Newton methods, quadratic programming, dynamic programming, simplex method, gradient method, etc., are the well-known classical methods to solve optimization problems[2,3]. These algorithms robustly result in the same solution for a given optimization problem with an identical initial starting point. Although such techniques can find optimal solutions in a reasonable time, they need the objective function and constraints to be convex and derivable[4]. These cause the deterministic algorithms to fall into locally optimal solutions, which is the main shortcoming of such methods in solving real-world problems. This defect becomes more prominent as the dimension of the problem increases. Therefore, stochastic methods for dealing with it have been developed[5–7]. These algorithms intrinsically neglect the characteristics of the objective functions and constraints, so they treat the problem as a black box. Another advantage of the most metaheuristic algorithms is their simplicity.

How metaheuristics solve a problem is similar. Evolve a set of random solutions in an iterative procedure and keep a good balance between exploration and exploitation phases[8]. To this end, the search process is divided into two stages exploration and exploitation[9]. The former assists an algorithm to search globally, and the latter helps it locally search around the probable region obtained by the former phase[10]. The common feature in metaheuristic algorithms is their randomness which can help them to avoid local solutions. However, these

[1]Artificial Intelligence Research Centre, Ajman University, Ajman, United Arab Emirates. [2]Department of Electrical Engineering, Faculty of Engineering, Jahrom University, Jahrom, Fars, Iran. [3]National Grid ESO, Warwick CV346DA, UK. [4]Centre for Artificial Intelligence Research and Optimisation, Torrens University Australia, Brisbane, Australia. [5]Yonsei Frontier Lab, Yonsei University, Seoul, South Korea. ✉email: m.deriche@ajman.ac.ae

features cause metaheuristic algorithms not to guarantee the global solution. Furthermore, they may result in different solutions in each run[11].

According to the No Free Lunch (NFL) theorem[12], there is no unique algorithm to solve all optimization problems effectively. This theorem motivates researchers to introduce new algorithms to be applied in various fields of study[13]. Many recently developed nature-inspired metaheuristic algorithms evidenced this theorem and encouraged us to introduce a novel nature-inspired algorithm based on cheetahs' hunting strategies.

Literature review. Metaheuristic algorithms can be categorized into two main classes, single-solution-based and multiple-solution-based (or population-based). The most popular single-solution-based metaheuristic algorithm is simulated annealing (SA)[14]. This algorithm's process starts with a random candidate solution (a population) and then moves and improves it in the promising search space in an iterative manner to find the superior solution. However, multiple-solution-based algorithms implement more than one random candidate solution to enhance the speed and the chance to avoid local optima entrapment.

Population-based metaheuristic algorithms can be classified into evolutionary, physics, chemistry, and swarm-based algorithms[15]. Most popular evolutionary-based metaheuristic algorithms are genetic algorithm (GA)[16], genetic programming (GP)[17], evolutionary programming (EP)[18], evolutionary strategy (ES)[19], biogeography-based optimizer (BBO)[20] and differential evolution (DE)[21]. The basis of population improvement and movement in these algorithms is derived from the concept of evolution in nature.

Physics-based metaheuristic algorithms are another type of population-based optimization algorithm. The improvement and movement of the population through search space in these algorithms is made by directly deploying the known laws of physics. These rules include the laws of mechanics, relativity, gravity, electrodynamics, electromagnetism, optics, etc.[15]. The most famous of these algorithms only cover those which have at least 100 citations, as measured by Google Scholar (collected in March 2021) are, respectively: gravitational search algorithm (GSA)[22], big bang-big crunch (BB-BC)[23], charged system search (CSS)[24], the electromagnetism-like mechanism (EM)[25], water cycle algorithm (WCA)[26], extremal optimization (EO)[27], ray optimization (RO)[28], central force optimization (CFO)[29], intelligent water drops (IWD)[30], chaos optimization algorithm (COA)[31], galaxy-based search algorithm (GBSA)[32] and river formation dynamics algorithm (RFDA)[33].

Another class of population-based metaheuristic algorithms is chemistry-based algorithms. These algorithms are created based on molecules' chemical reactions and features. The most popular chemistry-based algorithms are artificial chemical reaction optimization algorithm (ACROA)[34], gases Brownian motion optimization (GBMO)[35] and artificial chemical process (ACP)[36].

The most popular class of population-based metaheuristic algorithms for researchers is swarm-based algorithms. This type of algorithm is a model of the behavior and social intelligence of a group of living things, such as birds, ants, swarms, schools and so on[15]. Some of the most popular algorithms in this category are particle swarm optimization (PSO) in 1995[37], ant colony algorithms (ACO) in 1991[38], artificial bee colony (ABC) in 2007[39], Cuckoo Search (CS) in 2009[40], grey wolf optimizer (GWO) in 2014[41], firefly algorithm (FA) in 2009[42], bacterial foraging algorithm (BFA) in 2002[43], whale optimization algorithm (WOA) in 2016[44], bat algorithm (BA) in 2010[45], Shuffled frog leaping algorithm (SFLA) in 2003[46], bees algorithm (BA) in 2006[47], moth-flame optimization (MFO) in 2015[48], krill herd (KH) in 2012[49], ant lion optimizer (ALO) in 2015[50], fruit fly optimization algorithm (FOA) in 2012[51], and glowworm swarm optimization (GSO) in 2009[52].

Motivation. Some swarm intelligence algorithms are based on animals' hunting and foraging behaviors in nature. Some hunters can hunt the prey individually or in a herd with some numbers, and other members may not participate in the hunting process. Furthermore, in some cases, a small number of hunters can cover a large hunting area. These special features of the cheetah for hunting motivated us to study its behavior more carefully and base it on the development of an optimization algorithm. The hunting processes are modeled in two simple sit and wait beside the attacking mode strategies. Indeed, despite other methods which use some complicated equations in the evolution process, the cheetah optimizer (CO) employs some simple techniques, while the hunting strategies help increase the algorithm's effectiveness. Sitting and waiting to make the prey available, back to home in case of unsuccessful hunting process, return to last successful hunting if the prey not found for sometimes. These are the main strategies in CO. The algorithm performance confirms that the hunting process's characteristics have been modeled in the proposed CO.

Contributions. The main contributions of this paper are listed as follows:

1. A new population-based metaheuristic called CO algorithm is investigated, formulated, and tested on different benchmark functions.
2. Since the hunting processes of cheetahs, i.e., search, sit-and-wait, attack, and leave the prey and go back home, are modeled wholly and simply; thus, the number of initial populations are decreased dramatically in small to large scale optimization problems.
3. The proposed CO method needs a small number of equations while the hunting strategies try to model the hunting process. These strategies create a suitable trade-off between the exploration and exploitation searches and prevent premature convergence in the different optimization problems. Therefore, the concepts of proposed strategies can be used to enhance the performance of other metaheuristic algorithms effectively.

It should be noted that there are existing algorithms in the literature inspired by cheetah[53–55]. In[53], it has been tried to use the cheetahs' feature in the chasing mode as an optimization algorithm, but the model is not well shown conceptually and mathematically. In[54], an algorithm based on the hunting behavior of cheetahs was introduced. The model has been established based on GWO and modified to adapt to the group hunting behavior of cheetahs, such as leadership hierarchy and communication between teammates during the hunting. However, these algorithms have not been able to make good use of all the features and strategies of cheetahs during hunting and model them mathematically. This article tries to cover these shortcomings well to present a more realistic behavior of cheetahs in creating a robust meta-heuristic algorithm.

The rest of this paper is organized as follows. The inspiration for the suggested method is briefly addressed in "Inspiration" section. "Mathematical model and algorithm" section presents the proposed CO algorithm's mathematical model. "Simulation results" section presents experimental findings and discussions for various benchmark test functions and economic load dispatch (ELD) problems. Finally, concluding remarks and future work are summarized in "Conclusion" section.

Inspiration

Cheetah (Acinonyx jubatus) is the primary cat breed and fastest land animal living in the central areas of Iran and Africa[56]. The cheetah's speed can reach over 120 km per hour. The cheetahs' agility and speed are their physical characteristics like a long tail, long and thin legs, lightweight and flexible spine. Cheetahs are quick animals capable of stealthy movement, fast returning during predation, and specific spotted coats; however, these visual predators cannot maintain their high-speed action for a long time. Therefore, the chasing must be less than half of a minute[57].

Moreover, their speed significantly decreases from 93 km/h or 58 mph to 23 km/h 14 mph only in three strides after catching the prey. Due to the mentioned limitation of cheetahs in maintaining their speed, they precisely observe the environment after staying on small branches or hills to identify their prey. Furthermore, these big cats can effortlessly blend into the high and dry grass due to their specific coats[58].

These predators usually hunt gazelles, specifically Thomson's gazelles, impalas, antelopes, hares, birds, rodents, and calves of more fabulous herd animals. First, they move slowly toward the prey with a crouched posture to be hidden and reach the minimum distance, stopping hidden and waiting for the prey to approach the predator. This is because they stop hunting if the prey observes the predator. The mentioned minimum distance is almost 60–70 m or 200–230 ft; however, it is determined to be 200 m or 660 ft if they cannot stay hidden appropriately. Specifically, the pursuit duration is 60 s with a mean distance of 173 m or 568 ft to 559 m or 1834 ft. Then, the prey's balance is lost after their rump is beaten with the cheetah's forepaw, and finally, the predator brings down the prey using too much force and turns it, which makes the prey try to escape[59]. Cheetahs' muscular tails' back and forth movement also helps them achieve sharp turns[60]. Generally, hunting the animals that move far from their herds or have less caution is much easier[61,62]. It should be noted that there are various determinants associated with predation, including maturity, gender, the number of predators, and the carelessness of prey. Also, coalitions or mothers with cubs tend to hunt more giant animals successfully.

According to the biological investigations, it has been found that cheetahs have remarkable spinal flexibility and long tails that lead to their physical balance. Moreover, they have collarbone-separated shoulder blades that facilitate the movement of the shoulders. All the characteristics mentioned earlier make these big cats considered remarkable predators; however, not all the predations are successful.

Mathematical model and algorithm

When a cheetah is patrolling or scanning its surroundings, it is possible to detect prey. Seeing the prey, the cheetah may sit in its place and wait until the prey gets closer to it and then starts the attack. The attack mode includes rushing and capturing phases. The cheetah may give up the hunting for several reasons, such as its energy limits, fast prey fleeing, etc. Then, they may go back home to rest and start new hunting. By assessing the prey, his/her condition, area and distance to the prey, the cheetah may choose one of these strategies, as depicted in Fig. 1[63]. Overall, the CO algorithm is based on intelligently utilizing these hunting strategies during hunting periods (iterations).

- *Searching*: Cheetahs need to search, including scanning or active search, in their territories (search space) or the surrounding area to find their prey.
- *Sitting-and-waiting*: After the prey is detected, but the situation is not proper, cheetahs may sit and wait for the prey to come nearer or for the position to be better;
- *Attacking*: This strategy has two essential steps:
    - Rushing: When the cheetah decides to attack, they rush toward the prey with maximum speed.
    - Capturing: The cheetah used speed and flexibility to capture the prey by approaching the prey.
- *Leave the prey and go back home*: Two situations are considered for this strategy. (1) If the cheetah is unsuccessful in hunting the prey, it should change its position or return to its territory. (2) In cases with no successful hunting action in some time interval, it may change its position to the last prey detected and searched around it.

The mathematical models of the hunting mentioned above strategies are provided in detail in the following sections. Then, the CO is outlined.

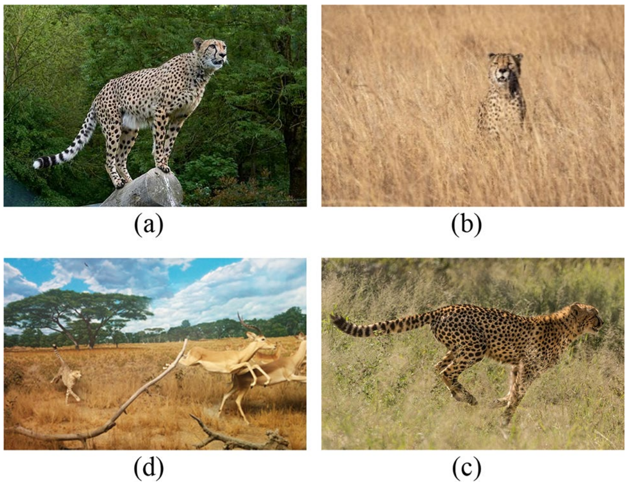
Figure 1. Hunting behavior of cheetahs: (**a**) searching for prey (scanning mode), (**b**) sitting-and-waiting (hiding), (**c**) rushing and (**d**) capturing.

Search strategy. Cheetahs seek prey in two ways; either scan the environment while sitting or standing or actively patrols the surrounding area. Scanning mode is more suitable when the prey is dense and grazing while walking on the plains. On the other hand, choosing an active mode that needs more energy than the scan mode is better if the prey is scattered and active. Therefore, during the hunting period, depending on the prey's condition, the coverage of the area, and the condition of the cheetahs themselves, a chain of these two search modes may be selected by the cheetah. To mathematically model this searching strategy of cheetahs, let $X_{i,j}^{t}$ denote the current position of cheetah i ($i=1, 2, \ldots, n$) in arrangement j ($j=1, 2, \ldots, D$), where n is the number of cheetahs population and D is the dimension of the optimization problem. Indeed, each cheetah experiences different situations dealing with various prey. Each prey is a location of a decision variable corresponding to the best solution while the cheetah's states (other arrangements) construct a population.

Then, the following random search equation is proposed for updating the new position of cheetah i in each arrangement based on their current position, and an arbitrary step size as follows:

$$X_{i,j}^{t+1} = X_{i,j}^{t} + \hat{r}_{i,j}^{-1}.\alpha_{i,j}^{t} \tag{1}$$

where $X_{i,j}^{t+1}$ and $X_{i,j}^{t}$ are the next and the current positions of cheetah i in arrangement j, respectively. Index t denotes the current hunting time, and T is the maximum length of hunting time. $\hat{r}_{i,j}^{-1}$ and $\alpha_{i,j}^{t}$ are the randomization parameter and step length for cheetah i in arrangement j, respectively. The second term is the randomization term, where the randomization parameter $\hat{r}_{i,j}$ is normally distributed random numbers from a standard normal distribution. The step length $\alpha_{i,j}^{t} > 0$ in most cases can be set at $0.001 \times t/T$ as cheetahs are slow-walking searchers. In encountering other hunters (enemies), the cheetahs may escape rapidly and change their directions. To reflect such behavior as well as near/far destination search mode, the random number $\hat{r}_{i,j}^{-1}$ is used here for each cheetah in different hunting periods. In some cases, $\alpha_{i,j}^{t}$ can be adjusted by the distance between the cheetah i and his/her neighborhood or leader. The position of a cheetah (named leader) in each arrangement of cheetahs is updated by assuming $\alpha_{i,j}^{t}$ equal to $0.001 \times t/T$ multiplied by the maximum step size (here, we consider it based on the variable limits, i.e., the upper limit minus the lower limit). For other members, $\alpha_{i,j}^{t}$ in each cheetah's arrangement is calculated by multiplying the distance between the position of cheetah i and a randomly selected cheetah. Figure 2a illustrates the search strategy.

There is a distance between the leader and the prey (i.e., the best solution in this paper). Thus, the leader position is selected based on the prey position by changing some variables in the best solution. It is to be expected that the leader and the prey would be closer by the time unless the hunting time is terminated, leading to an updated leadership position. It should be noted that the step size of the cheetah is entirely random, and the CO would consider this. Therefore, the CO can effectively solve the optimization problems correctly by utilizing any randomization parameter and random step size, i.e.,$\hat{r}_{i,j}^{-1}$ and $\alpha_{i,j}^{t}$.

Sit-and-wait strategy. During the searching mode, the prey may expose to a cheetah's field of vision. In this situation, every movement of the cheetah may make the prey aware of his/her presence and lead to the escape of the prey. To avoid this concern, the cheetah may decide to ambush (by lying on the ground or hiding among the bushes) to get close enough to the prey. Therefore, in this mode, the cheetah remains at his/her position and waits for the prey to come nearer (see Fig. 2b). This behavior can be modeled as follows:

$$X_{i,j}^{t+1} = X_{i,j}^{t} \tag{2}$$

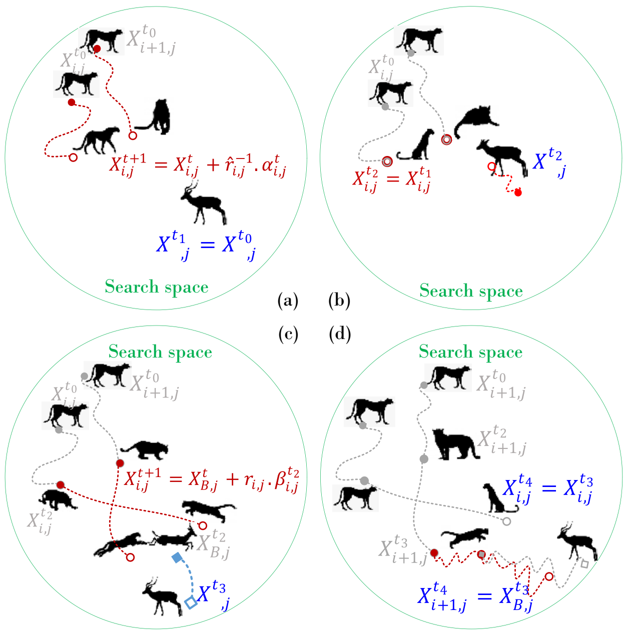
$X^{t_0}_{i+1,j}$
$X^{t_0}_{i,j}$
$X^{t+1}_{i,j} = X^{t}_{i,j} + \hat{r}^{-1}_{i,j}.\alpha^{t}_{i,j}$
$X^{t_1}_{,j} = X^{t_0}_{,j}$
Search space
(a)
(b)
$X^{t_0}_{i,j}$
$X^{t_2}_{i,j} = X^{t_1}_{i,j}$
$X^{t_2}_{,j}$
Search space
(c)
(d)
Search space
$X^{t_0}_{i,j}$
$X^{t_0}_{i+1,j}$
$X^{t+1}_{i,j} = X^{t}_{B,j} + r_{i,j}.\beta^{t_2}_{i,j}$
$X^{t_2}_{i,j}$
$X^{t_2}_{B,j}$
$X^{t_3}_{,j}$
Search space
$X^{t_0}_{i+1,j}$
$X^{t_2}_{i+1,j}$
$X^{t_4}_{i,j} = X^{t_3}_{i,j}$
$X^{t_3}_{i+1,j}$
$X^{t_4}_{i+1,j} = X^{t_3}_{B,j}$

Figure 2. Graphical information of CO's strategies.

where $X^{t+1}_{i,j}$ and $X^{t}_{i,j}$ are the updated and current positions of cheetah i in arrangement j, respectively. This strategy requires the CO algorithm not to change all cheetahs simultaneously in each group to increase the success of hunting (finding a better solution) and hence can assist it in avoiding premature convergence.

Attack strategy. Cheetahs use two crucial factors to attack their prey: speed and flexibility. When a cheetah decides to attack, he/she rushes to the prey at full speed. After a while, the prey notices the cheetah's attack and begins to flee. The cheetah rapidly pursues the prey in the interception path with its keen eyes, as shown in Fig. 2c. In other words, the cheetah follows the position of the prey and adjusts its direction of movement in such a way as to block the prey's way at one point. Because the cheetah has reached a short distance from the prey at maximum speed, the prey must escape and change its position suddenly to survive, as shown in Fig. 2d, i.e., the next position of the cheetah is near the last position of prey. Also, as shown in Fig. 2d, probably, the one cheetah doesn't participate in an attacking strategy that completely matches the natural hunting of cheetahs. The cheetah uses speed and flexibility to capture the prey in this phase. In a group hunting method, each cheetah may adjust his/her position based on the fleeing prey and the position of the leader or neighborhood cheetah. Simply, these all attacking tactics of cheetahs are mathematically defined as follows:

$$X^{t+1}_{i,j} = X^{t}_{B,j} + \check{r}_{i,j}.\beta^{t}_{i,j} \tag{3}$$

where $X^{t}_{B,j}$ is the current position of the prey in arrangement j. In other words, it is the current best position of the population. $\check{r}_{i,j}$ and $\beta^{t}_{i,j}$ are respectively the turning factor and interaction factor associated to the cheetah i in arrangement j. $X^{t}_{B,j}$ is used in (3) because, in attacking mode, the rushing strategy of cheetahs by utilizing maximum speed helps them get as close as possible to the prey's position in a short time. Hence, this paper calculates the new position of the i-th cheetah in attacking mode based on the prey's current position. In the second term, the turning factor $\beta^{t}_{i,j}$ reflects the interaction between the cheetahs or between a cheetah and leader in the capturing mode. Mathematically, this factor can be defined as the difference between the neighborhood cheetah's position, $X^{t}_{k,j}$ $(k \neq i)$, and the i-th cheetah's position, $X^{t}_{i,j}$. The turning factor $\check{r}_{i,j}$ is a random number that is equal to $|r_{i,j}|^{\exp(r_{i,j}/2)} \sin(2\pi r_{i,j})$ in this paper. $r_{i,j}$ is normally distributed random numbers from a standard normal distribution. This factor reflects the sharp turns of the cheetahs in the capturing mode.

Hypotheses. Based on the behaviors of cheetahs in haunting, the following assumptions are considered in the proposed CO algorithm:

1. Each row of the cheetahs' population is modeled as a cheetah in different states. Each column represents a specific arrangement of cheetahs concerning the prey (best solution of each decision variable). In other words, cheetahs follow their prey (best point of a variable). To find the best optimal solution, the cheetahs require success in capturing the prey in each arrangement. The performance of each cheetah is evaluated by the value of fitness function for that cheetah in all arrangements. The higher performance of a cheetah indicates a higher probability of hunting success.

2. In a real group hunting process, the reactions of all cheetahs are different from others. Indeed, in every arrangement, a cheetah may be in attacking mode while the other cheetahs may be in one of searching, sitting-and-waiting, and attacking modes. Also, the cheetahs' energy is independent of the prey. Modeling each decision variable as an arrangement of cheetahs, besides using the random parameters $\hat{r}_{i,j}^{-1}$ and $\check{r}_{i,j}$ results in preventing premature convergence even in an extremely large evolution process. These random variables can be considered an energy source for cheetahs during the hunting process. These two critical ideas have been neglected in previous hunting evolutionary methods that significantly affect optimization performance. In the attacking strategy, the cheetah's direction depends on the prey, but the cheetah movements have completely random behavior in the searching strategy.
3. The behaviors of cheetahs during the searching or the attacking strategies are assumed to be completely random, as depicted in Fig. 2 by the red dash-line, but during the rushing and the capturing mode, the prey scape in a sharp changing direction, as depicted in the last movement shown in Fig. 2d. The randomization parameter $\hat{r}_{i,j}$ and the turning factor $\check{r}_{i,j}$ model these random movements. Changing the step length $\alpha_{i,j}^{t}$ and interaction factor $\beta_{i,j}^{t}$ with completely random variables also lead to a suitable optimization process. These confirm that the hunting process is modeled precisely.
4. In the hunting process, the searching or the attacking strategy is deployed randomly, but the searching strategy becomes more likely over time due to decreasing the cheetah's energy level. In some cases, the first steps are devoted to the search strategy, while the attack strategy is selected for large values of t to achieve better solutions. Assuming r_2 and r_3 as uniformly random numbers from $[0, 1]$. If $r_2 \geq r_3$ the sit and wait strategy is selected; otherwise, one of the searching or the attacking strategies is selected based on a random value $H = e^{2(1-t/T)}(2r_1 - 1)$ where r_1 is a uniformly random number from $[0, 1]$. By tuning r_3, the switching rate between the sit-and-wait strategy and two other strategies can be controlled. For instance, based on our experiences, if the objective function is too sensitive to the changes in some decision variables (this can reflect the sensitivity of prey to the movement of the cheetah), the value of r_3 can be selected as small random numbers. This situation increases the sit-and-wait mode to be chosen by a cheetah, decreasing the rate of changing decision variables. Hence, the success probability of hunting (finding better solutions) is increased. Increasing t in the H function decreases the chance of choosing the attacking strategy by a cheetah due to the energy limitation. Still, this probability is not zero, which get entirely inspired by the cheetah's behavior. To do this, if $H \geq r_4$, attack mode is selected else the search mode is implemented. r_4 is a random number between 0 and 3. Here, higher values of the r_4 highlights the exploitation phase while decreasing it increases the exploration process.
5. The scanning and sitting-and-waiting strategies have the same meaning in the CO algorithm, indicating that the cheetah (search agent) has no movement during the hunting period.
6. If the leader fails in hunting in some consecutive hunting processing, the position of a randomly chosen cheetah is changed into the last place of success hunting (i.e., the prey position). Maintaining the prey position among a small population in this algorithm strengthens the exploration phase.
7. Each group of cheetahs has a limitation on the hunting time due to their energy limitations. Hence, if a group couldn't be succussed in a hunting period, the current prey is left, and the group comes back to its home range (initial position in this paper) to rest and then start another hunting. Indeed, a group of cheetahs will return home if their energies (which is modeled by hunting time) decrease and the leader's position is constant. In this condition, the position of leader is also updated. As a result of this strategy, it is possible to avoid getting stuck in local optimum solutions.
8. In each iteration, part of members participated in the evolution process.

Algorithm 1: *The CO Algorithm*

```
1:  Define the problem data, dimension ($D$), and the initial population size ($n$)
2:  Generate the initial population of cheetahs $X_i (i = 1, 2, \dots, n)$ and evaluate the fitness of each cheetah
3:  Initialize the population's home, leader and prey solutions
4:  $t \leftarrow 0$
5:  $it \leftarrow 1$
6:  $MaxIt \leftarrow$ desired maximum number of iterations
7:  $T \leftarrow 60 \times \lceil D/10 \rceil$
8:  while $it \leq MaxIt$ do
9:      Select $m$ $(2 \leq m \leq n)$ members of cheetahs randomly
10:     for each member $i \in m$ do
11:     Define the neighbor agent of member $i$
12:         for each arbitrary arrangement $j \in \{1, 2, \dots, D\}$ do
13:             Calculate $\hat{r}, \check{r}, \alpha, \beta$, and $H$
14:             $r_2, r_3 \leftarrow$ random numbers are chosen uniformly from 0 to 1
15:             if $r_2 \leq r_3$ then
16:                 $r_4 \leftarrow$ a random number is chosen uniformly from 0 to 3
17:                 if $H \geq r_4$ then
18:                     Calculate the new position of member $i$ in arrangement $j$ using Equation (3) //Attack
19:                 else
20:                     Calculate the new position of member $i$ in arrangement $j$ using Equation (1) //Search
21:                 end
22:             else
23:                 Calculate the new position of member $i$ in arrangement $j$ using Equation (2) //Sit-and-wait
24:             end
25:         end
26:         Update the solutions of member $i$ and the leader
27:     end
28:     $t \leftarrow t + 1$
29:     if $t > rand \times T$ and the leader position doesn't change for a time, then //Leave the prey and go back home
30:         Implement the leave the prey and go back home strategy and change the leader position
31:         Substitute the position of member $i$ by the prey position
32:         $t \leftarrow 0$
33:     end
34:     $it \leftarrow it + 1$
35:     Update the prey (global best) solution
36: end
```

The essential phases of the CO algorithm may be represented as the pseudo-code summarized in Algorithm 1 based on cheetah hunting techniques and assumptions. The source codes of this algorithm can be found in Supplementary Information.

Algorithm complexity. When evaluating the performance of an algorithm, its complexity is an important metric. In order to initialize each population in the CO as well as other algorithms such as PSO, GWO, GA, and DE, it requires $\mathcal{O}(o \times n)$ time, where o represents the number of objectives and n represents the number of populations. Every algorithm has an $\mathcal{O}(MaxIt \times ef)$ complexity, where $MaxIt$ is the maximum number of iterations and ef defines evaluation function complexity for a given problem. If the entire process is simulated, $\mathcal{O}(N)$ would require. Accordingly, algorithms such as PSO, and GWO have a computational complexity of $\mathcal{O}(N \times MaxIt \times o \times ef)$. For CO, GA and DE algorithms, the computational complexity is $\mathcal{O}(N \times MaxIt \times o \times n \times ef \times (mu + cr))$, where mu denotes mutation operations, and cr denotes crossover. Table 1 provides information on the average running time of CO and other algorithms for shifted sphere function with 10 variables for 1000 function evaluations and 25 independent runs. The population size for all algorithms is set at 10 and the other parameters of the competitor algorithms are given in Table 2. GWO is more time efficient compared to other approaches in terms of seconds. By contrast, GA exhibit relatively higher computational time than other competitors. The results of Table 1 indicate that, the CO can find better optimal solution in terms of mean and standard deviation (SD) than other algorithms in a reasonable time. It should be noted that the computational time also depends on the coding method of each algorithm. We use typical source codes in this comparison.

Simulation results

Results of CO on 500-D shifted CEC2005 test functions.

In this benchmark, f_1 to f_7 are classical unimodal functions, and f_8 to f_{13} are multimodal functions. This case study is designed to analyze the CO behavior compared to well-known algorithms such as DE, GWO, GA, PSO, and TLBO in large-scale 500-D shifted benchmark functions. In Table 2, the parameters for CO and other competing algorithms are shown. Since the population number differs from one case to another, the maximum number of function evaluations is set to 12×10^5 for

Algorithm	Mean	SD	Average time (s)
CO	4.12	20.01	1.45
PSO	760.37	1212.30	3.16
GA	2942.00	1550.60	4.60
DE	330.97	151.59	1.27
GWO	775.02	662.53	0.75
TLBO	289.37	406.04	1.77

Table 1. Time comparison of some metaheuristic algorithms performance.

all algorithm to have a firm comparison. The statistical results of this case are presented in Table 3. These results confirm the superiority of CO algorithm over the competing algorithms for nine test functions f_1, f_3, f_4, f_6, f_7, f_9, f_{10}, f_{12}, and f_{13} in terms of mean and standard deviation. For test functions f_2, and f_{11}, DE algorithm is better than the other algorithms. GA provides better solution for test function f_8, and TLBO shows better solution for test function f_5. Also, according to Freidman's rank test, the proposed CO algorithm shows the best performance among different competitor algorithms.

One of the major features of CO algorithm is the high capability in the exploitation phase, where the proposed strategies keep the population diversity and search the whole feasible search space, even in large scale optimization problem with small size of initial population, i.e., $n=6$. Indeed, in all objective functions, the population moves towards better solutions in consecutive iterations without trapping in local optima.

Results of CO on shifted-rotated CEC2005 test functions.

The set of benchmarks with 14 shifted-rotated CEC2005 functions with 30 variables are tested in this section. The results of CO are compared with several powerful, well-known evolutionary algorithms such as whale optimization algorithm (WOA)[44], emperor penguin optimizer (EPO)[64], slime mould algorithm (SMA)[65], Jaya[66], heat transfer search (HTS)[67], modified particle swarm optimizer (MPSO)[68], self-adaptive DE (jDE)[69], DE[70], and global and local real-coded genetic algorithms based on parent-centric crossover operators (GL-25)[71]. The parameters of each algorithm are reported in Table 4 and for a fair comparison the number of fitness evaluations are set to 3×10^5 for all algorithms.

The performance of CO is analyzed through the searching history, convergence, and trajectory curves. Also, some aspects of CO are cleared using the curves of minimum, maximum and mean value of the objective function in each iteration. Table 5 compares the statistical results of CO with nine evolutionary algorithms. As can be observed from Table 5, the proposed CO algorithm in all functions provides better solution than the WOA, EPO, SMA, Jaya, HTS, DE, and jDE algorithms. Also, in all test functions except f_{11}, CO algorithm overcomes the MPSO algorithm. Compared to GL-25, CO algorithm has been able to get better solutions for nine functions, i.e., functions f_2, f_3, f_7, f_8, f_9, f_{11}, f_{12}, f_{13}, and f_{14}. By contrast, for f_1, f_4, f_5, f_6, f_{10} GL-25 is better than other algorithms. Mean rank values from Freidman test reveal that CO algorithm has the best performance among the competitor algorithms for solving 30-D shifted-rotated CEC2005 test functions.

The plotted search history in Fig. 3 shows that the CO gains a better solution through exploring the feasible search space. As shown in Fig. 3, each variable changes its location from the lower limit to its upper limits several times, which guarantees to find suitable solutions. This regular searching history is a unique feature in the CO. The first graph from the right shows the minimum, maximum and average values of the objective function corresponding to the population in each iteration. Also, Fig. 3 shows that the leave the prey and go back home strategy keeps the population diversity during the evolution process. The cheetah's strategies are adopted to proceed with the optimal solution despite population diversity in each step of evolution.

Results of CO on large scale optimization problems.

In this section, to investigate how the CO method deals with the large-scale optimization problem, it is tested on CEC-2010 and CEC-2013 large-scale functions to validate its performance. The results of CO in the original condition are compared to other well-known modified algorithms which are tuned to solve the large-scale optimization problems, including[72]: (1) SLPSO, (2) CSO, (3) DMS-L-PSO, (4) CCPSO2, (5) DECC-G, (6) MLCC, and (7) DECC-DG methods as presented in Table 6. The results show that the CO can effectively enhance the results of eight benchmark functions while revealing completely comparable results in other objective functions. In the case of large-scale optimization problems, the CO uses 6-population while 2 of them are called in each iteration. The maximum number of function evaluations for all algorithms are set to 3×10^6 for these problems.

A comparison study between CO algorithm with a new modified version of PSO named dynamic group learning distributed particle swarm optimization (DGLDPSO)[73] is done in Table 7. As can be seen, the CO algorithm can defeat DGLDPSO in 10 functions while almost other objectives reach comparable values. It is worth noting that the CO can also be modified to achieve better solutions.

The CEC2013 large-scale benchmark functions[74] are selected as another case study. Implementing the CO algorithm on these benchmarks is compared with seven well-known modified optimization algorithms adopted to solve the large-scale optimization problems in Table 8. The CO algorithm enhances the results of ten benchmark functions which confirm the effectiveness of CO algorithm in dealing with large-scale optimization problems. The initial populations are set to 6, while 2-population is used in each iteration. Indeed, by changing the

Algorithm	Parameter	Value
CO	*N*	6
	m	2
DE	*n*	100
	CR	0.9
	F	0.9
GWO	*n*	40
GA	*n*	50
	Pc	0.95
	Pm	0.001
PSO	*n*	60
	w	0.9
	c_1, c_2	1.4961
TLBO	*n*	20

Table 2. Parameter setting of competing algorithms for solving 500-D shifted CEC2005 test functions.

turning factor and interaction factor ($\check{r}_{i,j}$ and $\beta^t_{i,j}$) and randomization parameter and step length ($\hat{r}^{-1}_{i,j}$ and $\alpha^t_{i,j}$) the results of other benchmark functions can be enhanced.

Results of CO on the real-life engineering application. The ELD problem is one of the most significant, well-known, and complicated optimization problems in power systems. This problem tries to determine the optimal output power generation of thermal units to meet the power system's required load demand while minimizing the units' fuel expenditures. ELD may also evaluate transmission system power losses and multi-fuel and valve point effects. The primary limitations of this problem include the restriction on load balance, restriction on power generation, restriction on the ramp rate, and restriction on prohibited operation zone.

The detailed ELD formulation is explained in[75]. a 15-unit test system is considered to evaluate the performance of CO algorithm to solve this nonconvex optimization issue. This system's total demand is 2630 MW. Other information about this system can be found in[75]. The results of statistical comparisons between CO and different meta-heuristic and improved algorithms in recent studies are summarized in Table 9. The competitive modified and hybrid algorithms in these two cases are: bacterial foraging optimization (BFO)[76], a modified ion motion optimization (MIMO) conglomerated with crisscross search (CSO) named C-MIMO-CSO[77], TLBO[78], an improved orthogonal design particle swarm optimization (IODPSO) algorithm[79], synergic predator–prey optimization (SPPO) algorithm[80], multi-strategy ensemble biogeography-based optimization (MsEBBO)[81], a new variant for the firefly algorithm, considering a non-homogeneous population named NhFA-Rnp[82], clustering cuckoo search optimization (CCSO)[83], a novel variant of competitive swarm optimizer (CSO) referred to as OLCSO[84], and adaptive charged system search (ACSS)[85]. The results demonstrate that the algorithm outperforms other state-of-the-art and improved algorithms in terms of worst, mean, best, and standard deviation values.

Conclusion

We proposed in this paper an optimization algorithm named cheetah optimizer (CO) based on the hunting process of cheetahs in nature. The proposed algorithm relies on several hunting strategies used by cheetahs instead of using mathematically complex approaches. In this regard, each decision variable is considered a possible arrangement of a group of cheetahs. Hence, each population can be regarded as a probable arrangement of cheetahs. The search, sit-and-wait and attack were mathematically modeled as the primary strategies of the proposed CO algorithm. Leave the prey and back to the home strategy was also implemented to enhance the algorithm's abilities in avoiding premature convergence and local optimal entrapment. These concepts were modeled in the CO framework so that it became an easy, fast, and powerful evolutionary method. The experimental results confirmed the monotonic behavior of the CO algorithm in dealing with low- and large-scale optimization problems. Finally, we validated the performance of the CO algorithm over the practical nonconvex ELD problem. The results showed that the suggested algorithm outperformed existing state-of-the-art algorithms in solving complex and challenging optimization problems. The main direction for future works is the development of the multi-objective CO, the application of CO to some complex engineering problems, and the hybridization of the proposed hunting strategies with other evolutionary methods.

FUN	Properties	CO	DE	GWO	GA	PSO	TLBO
f_1	Mean	**8.46E−04**	1.29E−03	8.31E+05	3.16E+01	5.74E+05	1.37E+06
	SD	**3.90E−05**	2.21E−04	3.55E+04	8.01E+00	4.56E+04	3.66E+04
f_2	Mean	4.81E−01	**1.14E−04**	1.55E+03	7.17E+01	1.05E+03	2.99E+03
	SD	1.32E−02	**1.39E−05**	6.31E+01	2.95E+00	7.80E+01	1.34E+02
f_3	Mean	**1.60E+05**	8.80E+06	2.73E+06	8.16E+05	5.17E+06	8.56E+06
	SD	**1.04E+04**	4.38E+05	1.98E+05	8.36E+03	7.42E+05	1.47E+06
f_4	Mean	**3.44E+01**	7.82E+01	9.37E+01	6.09E+01	1.31E+02	9.97E+01
	SD	**1.88E+00**	3.13E+00	8.16E−01	1.74E+00	7.72E+00	2.07E−01
f_5	Mean	1.06E+03	1.51E+04	3.21E+09	2.72E+03	2.74E+09	**4.98E+02**
	SD	2.28E+02	3.13E+03	2.10E+08	3.18E+02	5.35E+08	**1.36E−01**
f_6	Mean	**2.51E−01**	2.52E−01	7.88E+05	2.11E+01	5.79E+05	1.29E+06
	SD	**5.45E−05**	2.46E−04	4.68E+04	8.25E+00	3.74E+04	4.71E+04
f_7	Mean	**1.09E−01**	1.21E+00	2.29E+04	1.08E+00	2.04E+04	5.28E+04
	SD	**9.30E−03**	1.17E−01	1.14E+03	9.25E−02	4.22E+03	1.42E+03
f_8	Mean	−1.71E+05	−1.32E+05	−6.47E+04	**−2.09E+05**	−1.18E+05	−3.13E+04
	SD	3.13E+03	4.81E+03	4.07E+03	**2.45E+02**	4.45E+03	3.90E+03
f_9	Mean	**4.20E+01**	4.00E+03	6.06E+03	7.28E+02	6.25E+03	8.16E+03
	SD	**5.21E+00**	6.81E+01	1.73E+02	3.25E+01	3.34E+02	3.69E+02
f_{10}	Mean	**7.40E−03**	6.48E+00	1.96E+01	7.71E+00	2.04E+01	2.00E+01
	SD	**9.00E−04**	3.31E+00	1.31E−01	3.71E−01	3.77E−01	1.62E−03
f_{11}	Mean	3.20E−01	**1.59E−02**	6.74E+03	1.13E+00	4.83E+03	1.14E+04
	SD	3.37E−01	**4.95E−02**	4.54E+02	1.48E−01	6.10E+02	3.60E+02
f_{12}	Mean	**6.22E−04**	1.35E+06	7.35E+09	2.47E−03	7.58E+09	1.45E+10
	SD	**1.97E−03**	4.12E+06	6.13E+08	1.07E−03	3.04E+09	7.27E+08
f_{13}	Mean	**4.00E−01**	8.00E+04	1.32E+10	1.48E+00	1.23E+10	2.84E+10
	SD	**6.64E−06**	2.81E+04	1.27E+09	2.45E−01	1.63E+09	1.69E+09
Mean rank in Freidman test		**1.307692**	2.769231	4.615385	2.384615	4.538462	5.384615

Table 3. Results of 500-D shifted CEC2005 test functions. Significant values are in [bold].

Algorithm	Parameter value
CO	$n=6$, $m=2$
WOA	$n=30$, $a1\in[2\ 0]$; $a2\in[-2-1]$; $b=1$
EPO	$n=80$, $A\in[-1.5, 1.5]$, $T'\in[1, 1000]$, $S\in[0, 1.5]$, $M=2$, $f\in[2, 3]$, $l\in[1.5, 2]$
SMA	$n=30$, $z=0.03$, $a\in[-1, 1]$, $b\in[0, 1]$
Jaya	$n=50$, $r\in[0, 1]$
HTS	$n=50$, $R\in[0,1]$
MPSO	$n=40$, $w=0.9$, $c_1=c_2=1.4961$
GL-25	Not Available
DE	$n=100$, $CR=0.9$, $F=0.9$
jDE	$n=60$, $F_l=0.1$, $F_u=0.9$, $\tau_1=\tau_2=0.1$

Table 4. Parameter setting of competing algorithms for solving 30-D shifted-rotated CEC2005 test functions.

Algorithm			CO	WOA	EPO	SMA	Jaya	HTS	MPSO	GL-25	DE	jDE
Unimodal functions	f_1	Mean	5.70E−13	1.22E+03	8.47E+03	1.27E+01	4.86E+03	7.46E+01	6.05E+02	**5.60E−27**	7.69E+03	5.38E−12
		SD	1.76E−12	8.24E+02	5.29E+03	2.80E+01	2.39E+03	3.76E+01	2.24E+02	**1.76E−26**	1.65E+03	2.12E−12
	f_2	Mean	**3.25E−05**	5.87E+03	5.17E+03	3.93E+02	2.80E+03	1.49E+02	2.31E+03	4.04E+01	3.93E+04	1.23E+02
		SD	**1.27E−05**	1.95E+04	3.05E+03	1.75E+02	2.27E+03	9.37E+01	4.40E+02	6.28E+01	6.07E+03	2.00E+01
	f_3	Mean	**8.02E+05**	1.26E+08	2.29E+08	2.44E+07	4.50E+07	7.24E+07	1.61E+07	2.19E+06	1.04E+08	4.21E+06
		SD	**4.11E+05**	3.45E+08	6.21E+07	7.14E+06	8.32E+06	3.16E+07	1.04E+07	1.08E+06	4.00E+07	9.02E+05
	f_4	Mean	1.52E+03	6.27E+04	3.75E+03	1.68E+03	2.52E+04	1.62E+03	1.11E+04	**9.07E+02**	5.05E+04	5.99E+03
		SD	7.71E+02	5.77E+04	5.99E+02	1.90E+03	9.37E+03	1.21E+03	4.01E+03	**4.25E+02**	1.51E+04	1.36E+03
	f_5	Mean	5.82E+03	1.01E+04	1.89E+04	9.21E+03	7.54E+03	7.49E+03	6.62E+03	**2.51E+03**	1.13E+04	5.14E+03
		SD	1.31E+03	7.93E+03	9.25E+03	3.16E+03	2.55E+03	3.92E+03	2.25E+03	**1.96E+02**	1.42E+03	7.36E+02
Basic multimodal functions	f_6	Mean	5.10E+01	5.25E+08	3.76E+07	1.06E+04	1.21E+04	1.19E+06	2.14E+07	**2.15E+01**	5.89E+08	3.57E+01
		SD	6.61E+01	2.74E+08	2.43E+07	7.87E+03	6.16E+03	9.12E+05	2.38E+07	**1.17E+00**	3.41E+08	3.46E+00
	f_7	Mean	**2.06E−02**	3.59E+02	5.36E+02	5.47E+01	3.85E+00	3.76E+02	1.37E+02	2.78E−02	4.70E+03	6.22E−02
		SD	**1.71E−02**	2.60E+01	4.29E+02	1.25E+01	1.43E+00	3.02E+02	7.92E+01	3.62E−02	2.73E+00	1.97E−02
	f_8	Mean	**2.02E+01**	2.13E+01	2.15E+01	2.11E+01	2.12E+01	2.13E+01	2.07E+01	2.09E+01	2.11E+01	2.09E+01
		SD	**7.71E−02**	1.15E−01	1.26E+00	9.93E−0	7.39E−01	9.17E−02	1.03E−01	5.94E−02	2.86E−02	4.59E−02
	f_9	Mean	**2.03E−10**	4.00E+02	5.34E+02	2.27E+02	3.02E+02	2.07E+02	1.65E+02	2.45E+01	2.66E+02	4.34E+01
		SD	**8.37E−10**	8.52E+01	4.29E+01	6.18E+01	9.38E+01	7.99E+01	3.73E+01	7.35E+00	2.48E+01	6.04E+00
	f_{10}	Mean	1.99E+02	6.16E+02	3.48E+02	5.39E+02	5.05E+02	4.39E+02	2.07E+02	**1.42E+02**	3.37E+02	1.92E+02
		SD	5.53E+01	7.20E+01	7.90E+01	2.04E+02	3.14E+02	2.72E+02	6.82E+01	**6.45E+01**	1.24E+01	1.35E+01
	f_{11}	Mean	2.69E+01	4.18E+01	6.14E+01	4.26E+01	5.96E+01	6.80E+01	**2.68E+01**	3.27E+01	4.27E+01	2.79E+01
		SD	3.26E+00	2.92E+01	1.19E+01	3.33E+01	4.47E+01	5.64E+01	**4.03E+00**	7.79E+00	7.98E−01	1.18E+00
	f_{12}	Mean	**1.76E+03**	2.35E+05	3.59E+05	1.29E+04	1.70E+05	6.00E+04	8.92E+04	6.53E+04	4.84E+05	3.78E+04
		SD	**2.25E+03**	5.79E+03	9.37E+04	8.14E+03	7.31E+04	3.18E+04	3.90E+04	4.69E+04	1.27E+05	5.63E+03
Expanded multimodal functions	f_{13}	Mean	**1.32E+00**	1.93E+01	1.70E+01	1.90E+01	1.87E+01	1.67E+01	1.08E+01	6.23E+00	2.47E+01	6.04E+00
		SD	**3.04E−01**	2.28E+00	1.60E+00	2.04E+00	1.04E+00	2.85E+00	1.64E+00	4.88E+00	1.50E+00	5.82E−01
	f_{14}	Mean	**1.28E+01**	1.49E+01	1.55E+01	1.60E+01	1.40E+01	1.38E+01	1.32E+01	1.31E+01	1.39E+01	1.29E+01
		SD	**3.32E−01**	8.49E−01	1.10E+01	7.19E−01	5.38E−01	6.47E−01	1.05E−01	1.84E−01	1.90E−01	1.56E−01
Mean rank in Freidman test			1.642857	8.321429	8.5	5.75	6.857143	5.892857	4.642857	2.25	8.25	2.892857

Table 5. Comparison results between CO and nine evolutionary algorithms on shifted rotated CEC2005 test functions. Significant values are in [bold].

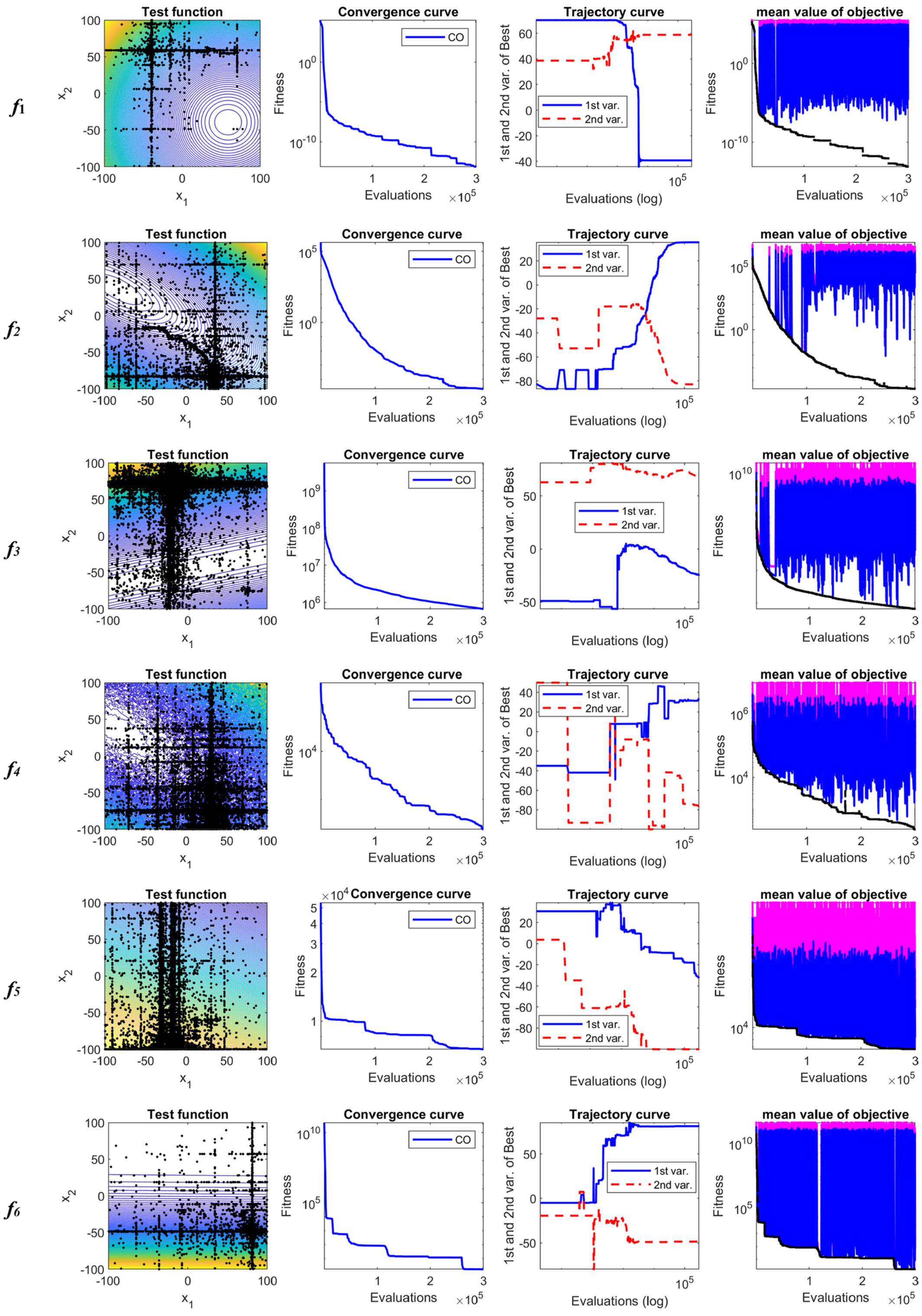

Figure 3. Convergence characteristics of CO on shifted-rotated CEC2005 benchmark functions.

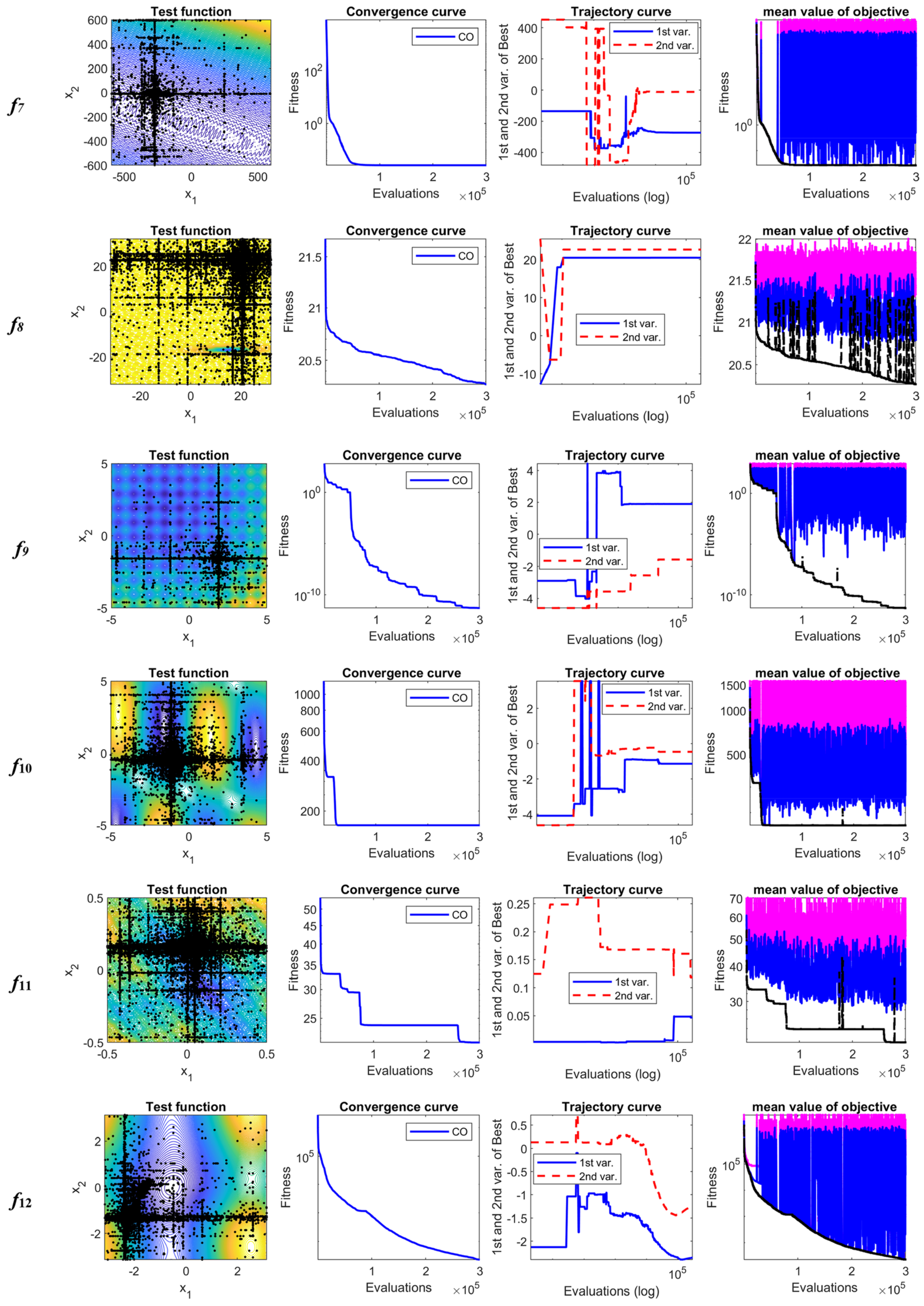

Figure 3. (continued)

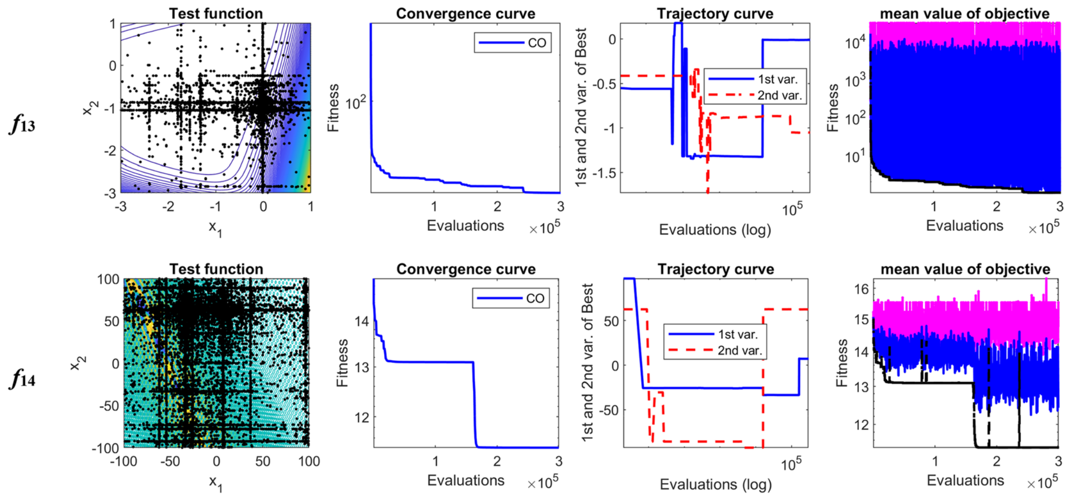

Figure 3. (continued)

FUN	Properties	CO	CSO	CCPSO2	DECC-DG	MLCC	DMS-L-PSO	DECC-G	SLPSO
f_{1}	Mean	2.4E−3	4.5E−12	1.6E+0	16.1E+0	**000.0E+0**	4.9E+9	13.6E−15	8.7E−18
	SD	179.0E−6	778.0E−15	1.7E+0	11.9E+0	**000.0E+0**	231.0E+6	1.7E−15	519.0E−21
f_{2}	Mean	115.3E+0	7.4E+3	7.5E+0	4.5E+3	**332.0E−3**	6.2E+3	48.8E+0	1.9E+3
	SD	12.6E+0	286.0E+0	1.7E+0	209.0E+0	**543.0E−3**	187.0E+0	13.1E+0	112.0E+0
f_{3}	Mean	14.1E−6	**2.6E−9**	8.9E−3	16.7E+0	81.7E−3	17.4E+0	1.7E+0	1.9E+0
	SD	273.0E−9	**452.0E−12**	11.7E−3	271.0E−3	311.0E−3	56.1E−3	288.0E−3	166.0E−3
f_{4}	Mean	**107.0E+9**	725.0E+9	1.2E+12	3.8E+12	15.4E+12	39.0E+12	12.5E+12	299.0E+9
	SD	**23.0E+9**	123.0E+9	741.0E+9	675.0E+9	6.6E+12	4.4E+12	2.9E+12	71.6E+9
f_{5}	Mean	279.0E+6	**11.5E+6**	453.0E+6	154.0E+6	313.0E+6	104.0E+6	260.0E+6	31.7E+6
	SD	61.6E+6	**1.6E+6**	118.0E+6	19.0E+6	109.0E+6	7.8E+6	81.6E+6	6.2E+6
f_{6}	Mean	8.5E+6	**821.0E−9**	19.2E+6	16.4E+0	16.1E+6	1.7E+6	4.8E+6	20.8E+0
	SD	1.8E+6	**46.1E−9**	1.7E+6	273.0E−3	4.4E+6	442.0E+3	614.0E+3	4.0E+0
f_{7}	Mean	**92.8E−6**	20.1E+3	170.0E+6	5.8E+3	1.8E+6	24.2E+9	8.4E+6	64.9E+3
	SD	**15.0E−6**	3.9E+3	323.0E+6	2.7E+3	2.9E+6	1.6E+9	7.6E+6	56.0E+3
f_{8}	Mean	**4.2E+6**	38.7E+6	33.1E+6	39.4E+6	37.6E+6	143.0E+6	47.2E+6	7.8E+6
	SD	**11.4E+6**	68.1E+3	29.7E+6	29.8E+6	32.7E+6	34.2E+6	30.8E+6	1.6E+6
f_{9}	Mean	**25.2E+6**	70.3E+6	114.0E+6	59.5E+6	119.0E+6	5.8E+9	254.0E+6	33.0E+6
	SD	**2.2E+6**	5.7E+6	36.0E+6	9.2E+6	14.4E+6	202.0E+6	10.1E+6	4.5E+6
f_{10}	Mean	4.9E+3	9.6E+3	5.7E+3	4.6E+3	3.0E+3	5.9E+3	9.2E+3	**2.6E+3**
	SD	175.0E+0	76.7E+0	1.0E+3	121.0E+0	370.0E+0	248.0E+0	437.0E+0	**217.0E+0**
f_{11}	Mean	194.0E+0	**40.2E−9**	198.0E+0	11.3E+0	196.0E+0	182.0E+0	25.2E+0	23.2E+0
	SD	1.5E+0	**6.9E−9**	239.0E−3	504.0E−3	3.1E+0	8.6E+0	1.1E+0	2.1E+0
f_{12}	Mean	**133.0E+0**	437.0E+3	27.8E+3	2.5E+3	36.0E+3	2.8E+6	39.1E+3	17.5E+3
	SD	**15.0E+0**	62.2E+3	7.6E+3	314.0E+0	6.5E+3	110.0E+3	5.8E+3	9.1E+3
f_{13}	Mean	1.2E+3	**629.0E+0**	1.3E+3	4.9E+3	2.4E+3	96.8E+6	3.1E+3	959.0E+0
	SD	610.0E+0	**232.0E+0**	182.0E+0	2.7E+3	1.6E+3	26.2E+6	1.2E+3	374.0E+0
f_{14}	Mean	**78.7E+6**	249.0E+6	322.0E+6	340.0E+6	324.0E+6	5.0E+9	577.0E+6	84.1E+6
	SD	**5.8E+6**	15.3E+6	146.0E+6	18.5E+6	19.7E+6	343.0E+6	21.8E+6	6.3E+6
f_{15}	Mean	9.7E+3	10.1E+3	10.2E+3	**5.8E+3**	7.2E+3	6.2E+3	9.8E+3	11.2E+3
	SD	446.0E+0	52.3E+0	890.0E+0	**60.2E+0**	1.1E+3	276.0E+0	2.7E+3	86.5E+0
f_{16}	Mean	391.0E+0	58.9E−9	397.0E+0	**723.0E−15**	381.0E+0	339.0E+0	85.0E+0	25.1E+0
	SD	1.6E+0	6.3E−9	464.0E−3	**48.9E−15**	47.7E+0	788.0E−3	10.4E+0	2.4E+0
f_{17}	Mean	**3.7E+3**	2.2E+6	141.0E+3	41.8E+3	156.0E+3	2.7E+6	163.0E+3	90.0E+3
	SD	**286.0E+0**	155.0E+3	58.1E+3	1.1E+3	10.3E+3	154.0E+3	9.6E+3	15.8E+3
f_{18}	Mean	2.6E+3	**1.7E+3**	2.9E+3	15.1E+9	6.8E+3	2.8E+9	9.0E+3	2.8E+3
	SD	761.0E+0	**522.0E+0**	373.0E+0	1.9E+9	6.0E+3	530.0E+6	1.1E+3	833.0E+0
f_{19}	Mean	**533.0E+3**	10.1E+6	1.4E+6	1.7E+6	1.3E+6	16.3E+6	733.0E+3	5.1E+6
	SD	**21.2E+3**	564.0E+3	89.0E+3	104.0E+3	105.0E+3	670.0E+3	46.1E+3	705.0E+3
f_{20}	Mean	1.6E+3	**1.1E+3**	2.0E+3	61.7E+9	2.0E+3	4.1E+9	3.5E+3	1.9E+3
	SD	142.0E+0	**159.0E+0**	208.0E+0	5.8E+9	196.0E+0	634.0E+6	245.0E+0	180.0E+0
+ (CO is less)			11	17	15	14	15	13	12
− (CO is greater)			**9**	1	5	5	5	6	8
= (is equal)			**0**	2	0	1	0	1	0

Table 6. Comparison of results of CO and state-of-the-art methods on the CEC2010 large-scale functions. Significant values are in [bold].

FUN	Properties	CO	DGLDPSO	FUN	Properties	CO	DGLDPSO
f_1	Mean	2.4E−3	**4.55E−21**	f_{11}	Mean	1.94E+2	**3.28E−13**
	SD	1.79E−04	**2.61E−22**		SD	1.52E+00	**1.23E−14**
f_2	Mean	**1.153E+2**	7.35E+02	f_{12}	Mean	**1.33E+2**	6.25E+04
	SD	**1.26E+1**	4.52E+1		SD	**1.5E+1**	3.55E+03
f_3	Mean	1.41E−05	**2.33E−13**	f_{13}	Mean	1.20E+03	**5.70E+02**
	SD	2.73E−07	**1.55E−14**		SD	6.10E+2	**1.83E+02**
f_4	Mean	**1.07E+11**	3.26E+11	f_{14}	Mean	**7.87E+07**	1.46E+08
	SD	**2.30E+10**	5.98E+10		SD	**5.84E+06**	8.59E+06
f_5	Mean	2.79E+08	**2.83E+07**	f_{15}	Mean	**9.73E+03**	1.05E+04
	SD	6.16E+07	**1.06E+06**		SD	**4.46E+2**	9.96E+01
f_6	Mean	8.51E+06	**4.14E−09**	f_{16}	Mean	3.91E+2	**3.58E−13**
	SD	1.75E+06	**1.39E−10**		SD	1.57E+00	**1.14E−14**
f_7	Mean	**9.28E−05**	2.43E+01	f_{17}	Mean	**3.71E+03**	1.84E+06
	SD	**1.50E−05**	1.00E+01		SD	**2.86E+2**	3.03E+05
f_8	Mean	**4.18E+06**	2.84E+07	f_{18}	Mean	2.578E+3	**1.97E+03**
	SD	**1.14E+07**	1.99E+5		SD	7.61E+2	**6.27E+02**
f_9	Mean	**2.52E+07**	4.47E+07	f_{19}	Mean	**5.33E+5**	5.66E+06
	SD	**2.16E+06**	3.44E+06		SD	**2.12E+4**	2.88E+05
f_{10}	Mean	4.86E+03	**2.05E+03**	f_{20}	Mean	1.563E+3	**1.25E+03**
	SD	1.75E+2	**1.26E+02**		SD	1.42E+2	**1.18E+02**

Table 7. Results of CO and DGLDPSO on the CEC2010 large-scale functions. Significant values are in [bold].

FUN	Properties	CO	CSO	CCPSO2	DECC-DG	MLCC	DMS-L-PSO	DECC-G	SLPSO
f_1	Mean	111.0E−6	3.7E−12	5.3E+0	1.3E+3	**81.0E−27**	5.7E+9	2.2E−12	10.9E−18
	SD	15.5E−6	1.2E−12	1.3E+0	4.0E+3	**311.0E−27**	144.0E+6	883.0E−15	2.6E−18
f_2	Mean	198.0E+0	7.0E+3	12.4E+0	12.8E+3	**8.3E+0**	12.4E+3	49.0E+0	2.1E+3
	SD	24.3E+0	356.0E+0	1.0E+0	462.0E+0	**5.6E+0**	269.0E+0	22.5E+0	136.0E+0
f_3	Mean	**20.0E+0**	21.6E+0	20.0E+0	21.4E+0	20.0E+0	21.4E+0	20.1E+0	21.6E+0
	SD	**3.8E−9**	5.4E−3	40.0E−6	15.2E−3	1.7E−3	19.2E−3	2.3E−3	14.5E−3
f_4	Mean	**659.0E+6**	12.6E+9	16.1E+9	52.4E+9	87.6E+9	900.0E+9	142.0E+9	4.4E+9
	SD	**277.0E+6**	1.9E+9	7.6E+9	33.6E+9	28.5E+9	37.8E+9	64.9E+9	948.0E+6
f_5	Mean	21.8E+6	862.0E+3	17.0E+6	5.8E+6	10.2E+6	5.5E+6	7.5E+6	**841.0E+3**
	SD	3.0E+6	21.3E+3	4.1E+6	349.0E+3	1.9E+6	419.0E+3	1.3E+6	**123.0E+3**
f_6	Mean	**1.0E+6**	1.1E+6	1.1E+6	1.1E+6	1.1E+6	1.0E+6	1.1E+6	1.1E+6
	SD	**19.1E+3**	1.1E+3	9.5E+3	904.0E+0	3.7E+3	4.0E+3	582.0E+0	1.5E+3
f_7	Mean	**61.9E+3**	7.6E+6	116.0E+6	835.0E+6	430.0E+6	3.6E+9	398.0E+6	1.6E+6
	SD	**15.0E+3**	1.4E+6	87.1E+6	766.0E+6	218.0E+6	261.0E+6	303.0E+6	705.0E+3
f_8	Mean	**25.4E+12**	350.0E+12	620.0E+12	4.6E+15	4.6E+15	6.8E+15	2.9E+15	103.0E+12
	SD	**16.5E+12**	35.9E+12	584.0E+12	525.0E+12	3.7E+15	1.4E+15	1.3E+15	36.2E+12
f_9	Mean	1.7E+9	**39.4E+6**	3.2E+9	500.0E+6	898.0E+6	505.0E+6	597.0E+6	82.5E+6
	SD	212.0E+6	**6.4E+6**	600.0E+6	22.0E+6	197.0E+6	26.6E+6	119.0E+6	10.6E+6
f_{10}	Mean	92.4E+6	94.1E+6	93.7E+6	94.6E+6	**92.2E+6**	93.1E+6	93.0E+6	92.5E+6
	SD	630.0E+3	149.0E+3	440.0E+3	34.7E+3	**384.0E+3**	311.0E+3	553.0E+3	1.7E+6
f_{11}	Mean	**14.2E+6**	358.0E+9	930.0E+9	23.5E+9	120.0E+9	496.0E+9	59.0E+9	933.0E+9
	SD	**8.9E+6**	9.8E+9	9.6E+9	13.2E+9	27.7E+9	40.1E+9	44.9E+9	9.0E+9
f_{12}	Mean	1.7E+3	**1.3E+3**	2.0E+3	163.0E+9	2.1E+3	4.4E+9	3.4E+3	1.8E+3
	SD	255.0E+0	**82.3E+0**	88.7E+0	16.1E+9	199.0E+0	834.0E+6	269.0E+0	174.0E+0
f_{13}	Mean	**3.9E+6**	806.0E+6	2.0E+9	19.8E+9	8.2E+9	116.0E+9	4.5E+9	465.0E+6
	SD	**1.2E+6**	102.0E+6	554.0E+6	6.1E+9	2.7E+9	10.5E+9	647.0E+6	317.0E+6
f_{14}	Mean	**17.9E+6**	5.2E+9	142.0E+9	18.6E+9	118.0E+9	1.3E+12	75.3E+9	328.0E+6
	SD	**2.0E+6**	2.9E+9	98.7E+9	9.4E+9	68.6E+9	159.0E+9	34.4E+9	517.0E+6
f_{15}	Mean	**775.0E+3**	17.4E+6	3.7E+6	9.5E+6	6.7E+6	1.6E+9	4.8E+6	78.7E+6
	SD	**30.6E+3**	653.0E+3	1.7E+6	984.0E+3	917.0E+3	618.0E+6	421.0E+3	8.5E+6
+ (CO is less)			11	17	13	9	13	12	12
− (CO is greater)			4	2	2	5	2	3	3
= (is equal)			0	1	0	1	0	0	0

Table 8. Comparison of results of CO and state-of-the-art methods on the CEC2013 large-scale functions. Significant values are in [bold].

Method	Worst	Mean	Best	SD
BFO[76]	32,784.502	32,976.8120	–	8.58E+01
C-MIMO-CSO[77]	32,701.210	32,701.2101	32,701.2200	5.80E−03
TLBO[78]	32,697.215	32,697.2151	32,697.2151	0.00E+00
IODPSO-L[79]	32,692.390	32,692.3958	32,692.3900	0.00E+00
SPPO[80]	32,708.000	32,732.0866	32,789.0000	1.80E+01
MsEBBO[81]	32,692.397	32,692.3973	32,692.3975	6.09E+05
NhFA-Rnp[82]	32,697.910	32,700.5600	32,709.9400	2.64E+00
CCSO[83]	32,706.640	32,706.6422	32,706.6400	7.00E−04
OLCSO[84]	32,692.3961	32,692.3981	32,692.4033	2.20E−03
ACSS[85]	32,761.3126	32,727.6967	32,678.1290	2.55E+01
CO	**32,678.1866**	**32,678.1465**	**32,678.0999**	**2.75E−02**

Table 9. Statistical results of CO and other metaheuristic methods in solving ELD of the 15-unit test system. Significant values are in [bold].

Data availability
Data sharing is not applicable—no new data is generated.

Received: 22 April 2022; Accepted: 6 June 2022
Published online: 29 June 2022

References

1. Sergeyev, Y. D., Kvasov, D. E. & Mukhametzhanov, M. S. On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. *Sci. Rep.* **8**, 1–9 (2018).
2. Luo, X., Liu, H., Gou, G., Xia, Y. & Zhu, Q. A parallel matrix factorization based recommender by alternating stochastic gradient decent. *Eng. Appl. Artif. Intell.* **25**, 1403–1412 (2012).
3. Lu, T. & Liu, S.-T. Fuzzy nonlinear programming approach to the evaluation of manufacturing processes. *Eng. Appl. Artif. Intell.* **72**, 183–189 (2018).
4. Koc, I., Atay, Y. & Babaoglu, I. Discrete tree seed algorithm for urban land readjustment. *Eng. Appl. Artif. Intell.* **112**, 104783 (2022).
5. Spall, J. C. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control* Vol. 65 (Wiley, 2005).
6. Cruz, Y. J. *et al.* Ensemble of convolutional neural networks based on an evolutionary algorithm applied to an industrial welding process. *Comput. Ind.* **133**, 103530 (2021).
7. Haber, R. E., Beruvides, G., Quiza, R. & Hernandez, A. A simple multi-objective optimization based on the cross-entropy method. *IEEE Access* **5**, 22272–22281 (2017).
8. Alba, E. & Dorronsoro, B. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Trans. Evol. Comput.* **9**, 126–142 (2005).
9. Wang, J.-S. & Li, S.-X. An improved grey wolf optimizer based on differential evolution and elimination mechanism. *Sci. Rep.* **9**, 1–21 (2019).
10. Lozano, M. & García-Martínez, C. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report. *Comput. Oper. Res.* **37**, 481–497 (2010).
11. Kirkpatrick, S. Optimization by simulated annealing: quantitative studies. *J. Stat. Phys.* **34**, 975–986 (1984).
12. Wolpert, D. H. & Macready, W. G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1**, 67–82 (1997).
13. Singh, P., Dhiman, G. & Kaur, A. A quantum approach for time series data based on graph and Schrödinger equations methods. *Mod. Phys. Lett. A* **33**, 1850208 (2018).
14. Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. Optimization by simulated annealing. *Science* **220**, 671–680 (1983).
15. Xing, B. & Gao, W.-J. *Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms* (Springer, 2014).
16. Holland, J. H. Genetic algorithms. *Sci. Am.* **267**, 66–73 (1992).
17. Koza, J. R. Genetic programming as a means for programming computers by natural selection. *Stat. Comput.* **4**, 87–112. https://doi.org/10.1007/BF00175355 (1994).
18. Yao, X., Liu, Y. & Lin, G. Evolutionary programming made faster. *IEEE Trans. Evol. Comput.* **3**, 82–102 (1999).
19. Rechenberg, I. *Evolutionsstrategie'94* (Frommann-holzboog, 1994).
20. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **12**, 702–713 (2008).
21. Storn, R. & Price, K. Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **11**, 341–359 (1997).
22. Rashedi, E., Nezamabadi-Pour, H. & Saryazdi, S. GSA: a gravitational search algorithm. *Inf. Sci.* **179**, 2232–2248 (2009).
23. Erol, O. K. & Eksin, I. A new optimization method: big bang–big crunch. *Adv. Eng. Softw.* **37**, 106–111 (2006).
24. Kaveh, A. & Talatahari, S. A novel heuristic optimization method: charged system search. *Acta Mech.* **213**, 267–289 (2010).
25. Birbil, Şİ & Fang, S.-C. An electromagnetism-like mechanism for global optimization. *J. Glob. Optim.* **25**, 263–282 (2003).
26. Eskandar, H., Sadollah, A., Bahreininejad, A. & Hamdi, M. Water cycle algorithm: a novel metaheuristic optimization method for solving constrained engineering optimization problems. *Comput. Struct.* **110**, 151–166 (2012).
27. Boettcher, S. & Percus, A. G. Optimization with extremal dynamics. *Complexity* **8**, 57–62 (2002).
28. Kaveh, A. & Khayatazad, M. A new meta-heuristic method: ray optimization. *Comput. Struct.* **112**, 283–294 (2012).
29. Formato, R. A. Central force optimization. *Prog. Electromagn. Res.* **77**, 425–491 (2007).
30. Hosseini, H. S. *IEEE Congress on Evolutionary Computation* 3226–3231 (IEEE, 2007).
31. Bing, L. & Weisun, J. Chaos optimization method and its application. *Control Theory Appl.* **14**, 613–615 (1997).
32. Shah-Hosseini, H. Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. *Int. J. Comput. Sci. Eng.* **6**, 132–140 (2011).
33. Rabanal, P., Rodríguez, I. & Rubio, F. in *International Conference on Unconventional Computation.* 163–177 (Springer).
34. Alatas, B. ACROA: artificial chemical reaction optimization algorithm for global optimization. *Expert Syst. Appl.* **38**, 13170–13180 (2011).
35. Abdechiri, M., Meybodi, M. R. & Bahrami, H. Gases Brownian motion optimization: an algorithm for optimization (GBMO). *Appl. Soft Comput.* **13**, 2932–2946 (2013).
36. Irizarry, R. LARES: an artificial chemical process approach for optimization. *Evol. Comput.* **12**, 435–459 (2004).
37. Kennedy, J. & Eberhart, R. in *Proceedings of ICNN'95-International Conference on Neural Networks.* 1942–1948 (IEEE).
38. Colorni, A., Dorigo, M. & Maniezzo, V. in *Proceedings of The First European Conference on Artificial Life.* 134–142.
39. Karaboga, D. & Basturk, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm J. *Glob. Optim.* **39**, 459–471 (2007).
40. Yang, X.-S. & Deb, S. in *World Congress on Nature & Biologically Inspired Computing (NaBIC).* 210–214 (IEEE, 2009).
41. Mirjalili, S., Mirjalili, S. M. & Lewis, A. Grey Wolf optimizer. *Adv. Eng. Softw.* **69**, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007 (2014).
42. Yang, X.-S. in *International Symposium on Stochastic Algorithms.* 169–178 (Springer).
43. Passino, K. M. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Syst. Mag.* **22**, 52–67 (2002).
44. Mirjalili, S. & Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **95**, 51–67 (2016).
45. Yang, X.-S. in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010).* 65–74 (Springer, 2010).
46. Eusuff, M. M. & Lansey, K. E. Optimization of water distribution network design using the shuffled frog leaping algorithm. *J. Water Resour. Plan. Manag.* **129**, 210–225 (2003).
47. Pham, D. T. *et al. Intelligent Production Machines and Systems* 454–459 (Elsevier, 2006).
48. Mirjalili, S. Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. *Knowl. Based Syst.* **89**, 228–249 (2015).
49. Gandomi, A. H. & Alavi, A. H. Krill herd: a new bio-inspired optimization algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **17**, 4831–4845 (2012).
50. Mirjalili, S. The ant lion optimizer. *Adv. Eng. Softw.* **83**, 80–98 (2015).
51. Pan, W.-T. A new fruit fly optimization algorithm: taking the financial distress model as an example. *Knowl. Based Syst.* **26**, 69–74 (2012).

52. Krishnanand, K. & Ghose, D. Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intell.* **3**, 87–124 (2009).
53. Goudhaman, M. Cheetah chase algorithm (CCA): a nature-inspired metaheuristic algorithm. *Int. J. Eng. Technol.* **7**, 1804–1811 (2018).
54. Saravanan, D., Paul, P. V., Janakiraman, S., Dumka, A. & Jayakumar, L. A new bio-inspired algorithm based on the hunting behavior of cheetah. *Int. J. Inf. Technol. Project Manag. (IJITPM)* **11**, 13–30 (2020).
55. Klein, C. E., Mariani, V. C. & dos Santos Coelho, L. Cheetah Based Optimization Algorithm: A Novel Swarm Intelligence Paradigm. In *ESANN* 685–690 (2018).
56. O'Brien, S. J., Johnson, W. E., Driscoll, C. A., Dobrynin, P. & Marker, L. Conservation genetics of the cheetah: lessons learned and new opportunities. *J. Hered.* **108**, 671–677 (2017).
57. Marker, L., Boast, L. K. & Schmidt-Küntzel, A. *Cheetahs: Biology and Conservation* (Academic Press, 2018).
58. Krausman, P. R. & Morales, S. M. Acinonyx jubatus. *Mamm. Species* **2005**, 1–6 (2005).
59. Estes, R. D. *The Behavior Guide to African Mammals: Including Hoofed Mammals, Carnivores* (Primates. University of California Press, 2012).
60. Patel, A. & Braae, M. in *IEEE/RSJ International Conference on Intelligent Robots and Systems.* 5506–5511 (IEEE, 2013).
61. Aarde, R. J. & Dyk2, A. Inheritance of the king coat colour pattern in cheetahs Acinonyx jubatus. *J. Zool.* **209**, 573–578 (1986).
62. Phillips, J. A. Bone consumption by cheetahs at undisturbed kills: evidence for a lack of focal-palatine erosion. *J. Mammal.* **74**, 487–492 (1993).
63. https://pixabay.com.
64. Dhiman, G. & Kumar, V. Emperor penguin optimizer: a bio-inspired algorithm for engineering problems. *Knowl. Based Syst.* **159**, 20–50 (2018).
65. Li, S., Chen, H., Wang, M., Heidari, A. A. & Mirjalili, S. Slime mould algorithm: a new method for stochastic optimization. *Future Gener. Comput. Syst.* **111**, 300–323 (2020).
66. Rao, R. Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *Int. J. Ind. Eng. Comput.* **7**, 19–34 (2016).
67. Patel, V. K. & Savsani, V. J. Heat transfer search (HTS): a novel optimization algorithm. *Inf. Sci.* **324**, 217–246 (2015).
68. Shi, Y. & Eberhart, R. in *IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360).* 69–73 (IEEE, 1998).
69. Qiang, J., Mitchell, C. & Qiang, A. in *IEEE Congress on Evolutionary Computation (CEC).* 4061–4068 (IEEE, 2016).
70. Fu, L., Zhu, H., Zhang, C., Ouyang, H. & Li, S. Hybrid harmony search differential evolution algorithm. *IEEE Access* **9**, 21532–21555 (2021).
71. Wang, Y., Cai, Z. & Zhang, Q. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans. Evol. Comput.* **15**, 55–66 (2011).
72. Jian, J.-R., Chen, Z.-G., Zhan, Z.-H. & Zhang, J. Region encoding helps evolutionary computation evolve faster: a new solution encoding scheme in particle swarm for large-scale optimization. *IEEE Trans. Evol. Comput.* **25**, 779–793 (2021).
73. Wang, Z.-J. *et al.* Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling. *IEEE Trans. Cybern.* **50**, 2715–2729 (2019).
74. Li, X. *et al.* Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. *Gene* **7**, 8 (2013).
75. Sun, J., Fang, W., Wang, D. & Xu, W. Solving the economic dispatch problem with a modified quantum-behaved particle swarm optimization method. *Energy Convers. Manag.* **50**, 2967–2975 (2009).
76. Panigrahi, B. & Pandi, V. R. Bacterial foraging optimisation: Nelder-Mead hybrid algorithm for economic load dispatch. *IET Gener. Transm. Distrib.* **2**, 556–565 (2008).
77. Kumar, M. & Dhillon, J. S. A conglomerated ion-motion and crisscross search optimizer for electric power load dispatch. *Appl. Soft Comput.* **83**, 105641 (2019).
78. Nee Dey, S. H. Teaching learning based optimization for different economic dispatch problems. *Sci. Iran.* **21**, 870–884 (2014).
79. Qin, Q., Cheng, S., Chu, X., Lei, X. & Shi, Y. Solving non-convex/non-smooth economic load dispatch problems via an enhanced particle swarm optimization. *Appl. Soft Comput.* **59**, 229–242 (2017).
80. Singh, N. J., Dhillon, J. & Kothari, D. Synergic predator-prey optimization for economic thermal power dispatch problem. *Appl. Soft Comput.* **43**, 298–311 (2016).
81. Xiong, G., Shi, D. & Duan, X. Multi-strategy ensemble biogeography-based optimization for economic dispatch problems. *Appl. Energy* **111**, 801–811 (2013).
82. Kapelinski, K., Neto, J. P. J. & dos Santos, E. M. Firefly algorithm with non-homogeneous population: a case study in economic load dispatch problem J. . *Oper. Res. Soc.* **72**, 519–534 (2021).
83. Yu, J., Kim, C.-H. & Rhee, S.-B. Clustering cuckoo search optimization for economic load dispatch problem. *Neural Comput. Appl.* **32**, 16951–16969 (2020).
84. Xiong, G. & Shi, D. Orthogonal learning competitive swarm optimizer for economic dispatch problems. *Appl. Soft Comput.* **66**, 134–148 (2018).
85. Zakian, P. & Kaveh, A. Economic dispatch of power systems using an adaptive charged system search algorithm. *Appl. Soft Comput.* **73**, 607–622 (2018).

Author contributions
Conceptualization: M.A.A. and M.Z.; methodology: M.A.A. and M.Z.; software: M.A.A. and M.Z.; validation: R.A., S.M. and M.D.; formal analysis: R.A., S.M. and M.D.; investigation: M.A.A. and M.Z.; resources: S.M. and M.D.; data curation: M.D.; writing—original draft preparation, M.A.A. and M.Z.; writing—review and editing: R.A., S.M. and M.D.; visualization: X.X.; supervision: S.M. and M.D.; project administration: M.D.; funding acquisition: M.D. All authors have read and agreed to the published version of the manuscript.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-022-14338-z.

Correspondence and requests for materials should be addressed to M.D.

Reprints and permissions information is available at www.nature.com/reprints.