Expert Systems With Applications 213 (2023) 119211

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Termite life cycle optimizer

Hoang-Le Minh [a,b], Thanh Sang-To [b], Guy Theraulaz [c], Magd Abdel Wahab [d,*], Thanh Cuong-Le [b,*]

[a] Soete Laboratory, Department of Electrical Energy, Metals, Mechanical Constructions, and Systems, Faculty of Engineering and Architecture, Ghent University, 9000 Gent, Belgium
[b] Center For Engineering Application and Technology Solutions, Ho Chi Minh City Open University, Ho Chi Minh City, Viet Nam
[c] Centre de Recherches sur la Cognition Animale (CRCA), Centre de Biologie Intégrative (CBI), Université de Toulouse, CNRS, UPS, Toulouse, France
[d] Faculty of Mechanical - Electrical and Computer Engineering, School of Engineering and Technology, Van Lang University, Ho Chi Minh City, Vietnam

ARTICLE INFO

Keywords:
CEC2005
Optimal engineering design
Optimization
Meta-heuristic
Stochastic optimization
Lévy flight
Termite life cycle optimizer

ABSTRACT

This paper introduces a novel bio-inspired meta-heuristic optimization algorithm, named termite life cycle optimizer (TLCO), which is based on both the life cycle of a termite colony and the modulation of movement strategies used by many animal species in nature. Termite colonies are comprised of three distinct castes: the workers, the soldiers and the reproductive termites. Each caste undertakes a set of specific tasks that ensure the growth and survival of the colony. TLCO mimics the activities of these three castes that are implemented in a mathematical model. The model is then used to find the global optimum in classic optimization problems. First, the behaviors of the workers, soldiers and reproductive termites are used to simulate a balance between the tasks of exploration and exploitation. Second, the initial population securely records the information obtained at each iteration and transmits it to workers and soldiers at the next iteration. This process is repeated until the global optimum is found with the smallest error. Besides, a new proposed function combined with Lévy flight is used to modulate the movement of termites that increases its flexibility. Thus, TLCO can cover both long distances during the first iterations to improve the convergence rate and shorter distances during the last iterations to enhance the level of accuracy. We then compare the performances of TLCO with other well-known nature-inspired algorithms using 23 classical benchmark functions, CEC2005 benchmark functions, and five real engineering design problems. The results demonstrate the effectiveness and reliability of TLCO in solving these optimization problems. Source codes of TLCO is publicly available at http://goldensolutionrs.com/termite-life-cycle-optimizer.html.

1. Introduction

Optimization algorithms allow us to find solutions for optimization problems (Haupt & Ellen Haupt, 2004). Depending on each problem, an objective function is first defined and then the maximum or minimum of the objective is determined by an optimization algorithm. An optimization problem can be defined in the following way:

Given: $f : A \rightarrow R$ from a set A to a real number.

Sought: $x_0 \in A$ such that $f(x_0) \leqslant f(x)$ for all $x \in A$ (minimization).

$x_0 \in A$ such that $f(x_0) \geqslant f(x)$ for all $x \in A$ (maximization).

Besides using the approaches of mathematics and numerical analysis, meta-heuristic algorithm is considered as an effective approach to solve optimization problems in various domains. Meta-heuristic is designed for solving a problem faster when traditional methods are too slow, or for searching the best solution with an acceptable error when classical methods fail to find the exact solution. This is achieved by repeating the process of "trial and error" continuously, and the experiences gained from the "error" solutions at the previous iteration will be recorded to adjust for the next iteration in a way that is suitable for the situation. The main characteristics of the *meta*-heuristic technique can be summarized as follows (Boussaïd, Lepagnot, & Siarry, 2013):

- Metaheuristic are strategies that lead the search process "trial and error".
- The goal is to explore the potential search space to find optimal solutions.
- Techniques that constitute metaheuristic algorithms range from simple search procedures to complex learning processes.
- Metaheuristic algorithms are approximate approach and are not problem-specific.

* Corresponding authors.
E-mail addresses: magd.a.w@vlu.edu.vn, magd.abdelwahab@ugent.be (M. Abdel Wahab), cuong.lt@ou.edu.vn (T. Cuong-Le).

https://doi.org/10.1016/j.eswa.2022.119211
Received 23 August 2021; Received in revised form 18 October 2022; Accepted 1 November 2022
Available online 11 November 2022

The process of "trial and error" for finding the global optima is secured by the number of solution candidates that is improved during optimization (the number of iterations). Based on the number of solutions, meta-heuristic optimization algorithms can be divided into two groups as single solution-based and population-based. There are several advantages and disadvantages for each groups. Single solution-based is less computationally costly, but suffers from early convergence, thus the level of accuracy is limited. On the contrary, population-based algorithms randomly generate a set of solution candidates $\vec{X_i} = \left\{\overrightarrow{X_1}, \overrightarrow{X_2}, .. \overrightarrow{X_n}\right\}$ in search space at the first step. Then, these candidates are combined and updated to explore and exploit the new search space $\vec{X_i}' = \left\{\overrightarrow{X_1'}, \overrightarrow{X_2'}, ... \overrightarrow{X_n'}\right\}$ through the process of repeat "trial and error". This will increase the opportunity for reaching the global optima. This group registers a high ability to avoid local optima since a set of solutions is involved during optimization, especially, with a large search space (Boussaïd, et al., 2013; Mahdavi, Shiri, & Rahnamayan, 2015; Poorzahedy & Rouhani, 2007). In addition, information sharing between solution candidates in coordination are improved in comparison with the first group and assist them to overcome different difficulties of search spaces. In other words, the solutions created in the next step will be more advanced than those in the previous step because of the useful information recorded. Besides these advantages, high computational cost and the need for more function evaluation are two major drawbacks of population-based algorithms.

Metaheuristic algorithms with different inspirations can be divided into three classes (Fister Jr, Yang, Fister, & Brest, 2013), i.e. evolution-inspired (Gong, Sun, & Ji, 2013; Mühlenbein, Gorges-Schleuter, & Krämer, 1988), physics-inspired (Biswas, Mishra, & Tiwari, 2013) and biological swarm-inspired (Krause, Cordeiro, Parpinelli, & Lopes, 2013). Evolution-inspired method is population-based approach and is inspired by the laws of biological systems (De Falco, Della Cioppa, Maisto, Scafuri, & Tarantino, 2012; Passino, 2002). The advantage of these algorithms is that each solution candidate is tied to the best solution found at the previous step. This allows the population to be optimized over the course of iterations. Genetic algorithm (GA) original version was proposed by Holland (Goldberg & Holland, 1988), which simulated the Darwinian evolution. GA techniques include mutation and crossover, to improve the solution candidate. The original version and its variants have been widely applied to many real-world problems (Gong, Sun, & Miao, 2016; Grefenstette, 2013; Tang, Man, Kwong, & He, 1996). Other popular algorithms were presented including Evolution Strategy (ES) (Beyer & Schwefel, 2002), Genetic Programming (GP) (Koza & Koza, 1992), Differential Evolution (DE) (Storn & Price, 1997), Evolutionary Programming (EP) (Juste, Kita, Tanaka, & Hasegawa, 1999), and Biogeography-Based Optimization algorithm (BBO) (Simon, 2008).

Physics-inspired method is inspired by physical phenomena in nature and is a population-based approach. In Simulated annealing (SA) algorithm (Kirkpatrick, Gelatt, & Vecchi, 1983), at each step iteration, SA registers some neighboring state s^* of the current state, and probabilistically decides whether to move the system to state s^* or to stay in-state s. These probabilities ultimately lead the system to move to state of lower energy. Typically, this step is repeated until the system reaches a state that is good enough for the application. Recently, many novel physics-inspired algorithms have been proposed including Gravitational Local Search (GLSA) (Webster & Bernhard, 2003), Gravitational Search Algorithm (GSA) (Rashedi, Nezamabadi-pour, & Saryazdi, 2009), Charged System Search (CSS) (Kaveh & Talatahari, 2010a, Kaveh & Talatahari, 2010b), Small-World Optimization Algorithm (SWOA) (Du, Wu, & Zhuang, 2006), Central Force Optimization (CFO) (Formato, 2007), Galaxy-based Search Algorithm (GbSA) (Shah-Hosseini, 2011), Black Hole (Hatamlou, 2013), Ray Optimization (RO) (Kaveh & Khayatazad, 2012a; Kaveh & Khayatazad, 2012b), curved space optimization (CSO) (Moghaddam, Moghaddam, & Cheriet, 2012), Atom search optimization (ASO) (Zhao, Wang, & Zhang, 2019) and so on.

The final class is Swarm-inspired algorithms. These algorithms mostly mimic the collective behavior of swarms of insects, herds of ungulates, flocks of birds, or schools of fish observed in nature (Camazine, et al., 2020; Moussaid, Garnier, Theraulaz, & Helbing, 2009). The mechanism is almost similar to physics-based algorithm, but the search is carried out by agents that navigate using the simulated collective intelligence specific to group-living species (Bonabeau, Marco, Dorigo, Théraulaz, & Theraulaz, 1999; Chakraborty & Kar, 2017; Patnaik, Yang, & Nakamatsu, 2017). These algorithms have become popular in solving optimization problems because of their strong global searching abilities. The background of these algorithms is based on simulating how to move, finding food, coordinating behavioral actions and sharing information among swarm particles. Hussain et al. (Hussain, Salleh, Cheng, & Shi, 2019) reported the trend preferred by researchers for designing new metaheuristic algorithms as shown in Fig. 1. Thus, the percentage of different animal groups used for simulating social characteristics, development and survival of swarms in nature is 48 % in total including 23 %, 16 % and 9 %. The popular algorithms that belong to these percentages are listed in Table 1.

It is clear that Swarm-inspired algorithms take an advantage when it is widely used for proposing a new algorithm as shown in Fig. 1. This is explained by the following reasons: (i) Simplicity is the primary advantage of Swarm-inspired algorithms, the majority of algorithms in this field follows a simple structure and has been inspired from simple concepts. This motivates a mathematical simulation to create different forms of swarm intelligence (SI) as given in Table 1. (ii) Swarm-inspired algorithms are population-based algorithms whose background is stochastic optimization algorithm, which is considered as black box (Droste, Jansen, & Wegener, 2006). This means that the process of derivation of the mathematical model is ignored. In other words, the optimization algorithms focus on changing the input and monitor the output for reaching the target (objective function).

Particle swarm optimization (PSO) (Kennedy & Eberhart, 1995) is the most popular swarm intelligence algorithm to solve continuous optimization problems. The main concept in PSO includes two factors. The first is to create a balance between two important features such as exploitation and exploration. According to (Dorigo, et al., 1996), exploration means the ability of the algorithm to find the new search space, which is far from the current particle position. And exploitation means the ability of the algorithm to find the potential position near the best position recorded. The second is that the information in PSO will be recorded after each iteration. Thus, the global best solution and the local best solution obtained at the previous iteration will be transmitted to each particle at the next iteration. This information will guide each particle to improve itself over the course of iterations. In PSO, the velocity is used as a separate strategy to move to a new position between local optima and global optima. In the mathematical form of velocity, the balance is established through two random vectors used to provide diversity to particle's movement and two parameters relative influence of the local best and global best. The new position of each particle will be updated by a combination between the current particle position and its velocity. This position will reach convergence when the number of iterations is sufficient.

The concept in PSO can be perceived as original and it is a rich source of inspiration for researchers to propose new algorithms in the past two decades. The new swarm algorithms almost are established by proposing different approaches through simulating swarm intelligence by mathematical models. Grey Wolf optimizer (GWO) (Mirjalili, et al., 2014) proposed a new technique called "encircling" and "attacking" to simulate the ability of exploration and exploitation. These abilities were done by a skillful vector, which showed either larger than one or smaller than one. Between each iteration, the shared information was secured by updating three best solutions in GWO. The advantage of GWO was that the updating of the new positions oriented well (near the potential location of the best solution) and not too far from the best solution. This

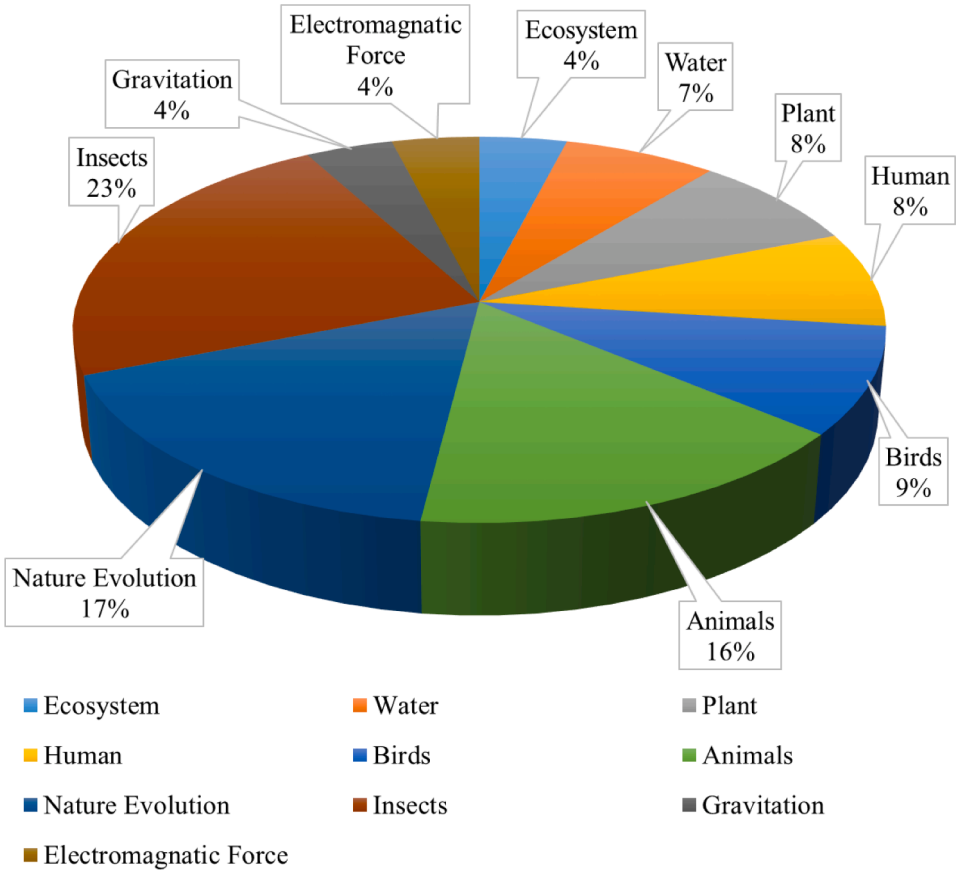

Fig. 1. Metaphors adopted by researchers for designing new metaheuristics.

Table 1
Optimization algorithms inpired by the behaviors of biological collectives proposed in literature.

Algorithm	Year of propose proposed
1. Particle swarm optimization (PSO) (Kennedy & Eberhart, 1995)	1995
2. Ant colony optimization (ACO) (Dorigo, Maniezzo, & Colorni, 1996)	1996
3. Bacterial foraging optimization (BFO)	2002
4. Artificial bee colony algorithm (ABC) (Basturk, 2006)	2005
5. Termite Algorithm (TA) (Roth, 2005)	2005
6. Glowworm swarm optimization (GSO) (Krishnanand & Ghose, 2009)	2005
7. Shuffled frog leaping algorithm (SFLA) (Eusuff, Lansey, & Pasha, 2006)	2006
8. Cat Swarm Optimization (CAT) (Chu, Tsai, & Pan, 2006)	2006
9. Bees algorithm (BA) (Pham, et al., 2006)	2006
10. Wasp Swarm Algorithm (WSO) (Pinto, Runkler, & Sousa, 2007)	2007
11. Monkey search (MA) (Mucherino & Seref, 2007)	2007
12. Wolf pack search algorithm (Yang, Tu, & Chen, 2007)	2007
13. Bee Collecting Pollen Algorithm (BCPA) (Lu & Zhou, 2008)	2008
14. Cuckoo search (CS) (Yang & Deb, 2009)	2009
15. Dolphin Partner Optimization (DPO) (Shiqin, Jianjun, & Guangxing, 2009)	2009
16. Bat algorithm (BA) (Yang, 2010b)	2010
17. Firefly Algorithm (FA) (Yang, 2010a)	2010
18. Hunting Search (HS) (Oftadeh, Mahjoob, & Shariatpanahi, 2010)	2010
19. Bird Mating Optimizer (BMO) (Askarzadeh & Rezazadeh, 2013)	2012
20. Krill Herd (KH) (Gandomi & Alavi, 2012)	2012
21. Fruit fly Optimization Algorithm (FOA) (Pan, 2012)	2012
22. Dolphin Echolocation (DE) (A. Kaveh & Farhoudi, 2013)	2013
23. The Smell Detection Agent (SDA) (Kaveh & Farhoudi, 2013)	2014
24. Grey Wolf optimizer (GWO) (Mirjalili, Mirjalili, & Lewis, 2014)	2014
25. The ant lion optimizer (ALO) (Mirjalili, 2015a)	2015
26. Dragonfly algorithm (Mirjalili, 2016a)	2016
27. The Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016)	2016
28. Killer Whale Algorithm (Biyanto, et al., 2017)	2016
29. Grasshopper optimization algorithm (GOA) (Saremi, Mirjalili, & Lewis, 2017)	2017
30. Salp Swarm Algorithm (SSA) (Mirjalili, et al., 2017)	2017
31. Emperor Penguins Colony (EPC) (Harifi, Khalilian, Mohammadzadeh, & Ebrahimnejad, 2019)	2019
32. A mayfly optimization algorithm (MA) (Zervoudakis & Tsafarakis, 2020)	2020
33. Horse herd Optimization Algorithm (HOA) (MiarNaeimi, Azizyan, & Rashki, 2021)	2021
34. Wild horse optimizer (WHO) (Naruei & Keynia, 2021)	2021
35. Honey Badger Algorithm (HBA) (Hashim, Houssein, Hussain, Mabrouk, & Al-Atabany, 2022)	2021

supports GWO to achieve a good convergence rate and a high accuracy level. Firefly Algorithm (FA) (Yang, 2010a) proposed a new technique of connection between each particle in swarm. The position updating in FA was oriented by the other particle, which was more attractive and a random vector was drawn from a Gaussian distribution. In FA algorithm, the exploration ability was represented by a random vector, while the exploitation was controlled by the attraction of different fireflies and the attractiveness strength. Especially, in case the light absorption coefficient that controls the decrease of light intensity is infinity, the FA algorithm will become an accelerated version of PSO. Bat algorithm (BA) [45] simulated the movement of Bats to detect prey, avoid obstacles, and locate their roosting crevices in the dark. Two parameters, called pulsed rate and loudness, proposed to select the way of position updating. The balance between exploration and exploitation was skillfully solved by comparing these parameters with a random scalar registered in the range from 0 to 1. The process of the velocities and positions updating in BA can be perceived as the same process in PSO (Kennedy & Eberhart, 1995). To a degree, BA can be considered a balanced combination of PSO and the intensive local search controlled by the loudness and pulse rate. The other algorithms have PSO characteristics such as Bird Mating Optimizer (BMO) [48], Gravitational Search Algorithm (GSA) (Rashedi, et al., 2009), Cuckoo search (CS) (Yang & Deb, 2009), Bacterial foraging optimization (BFO) (Passino, 2002) and so on.

The development of social technologies opens new challenges dealing with a great number of optimization problems. Although a large number of optimization algorithms has been introduced in the literature, new optimization algorithms are still being developed to solve emerging complex optimization problems and to obtain a better scheme (Minh, Sang-To, Wahab, & Cuong-Le, 2022; Sang-To, Hoang-Le, Wahab, & Cuong-Le, 2022). No Free Lunch Theorem of Optimization (Wolpert & Macready, 1997) proved that there is no optimization algorithm performing the overall best for different types of problems. This means that the success of any algorithm in solving a particular problem does not guarantee that it can solve efficiently other classes of problems. This theory promotes and encourages scholars to develop new algorithms or improve the current ones for solving the set of problems in the different fields (Cuong-Le, et al., 2021; Minh, Khatir, Rao, Abdel Wahab, & Cuong-Le, 2021; Minh, Khatir, Wahab, & Cuong-Le, 2021; Sang-To, et al., 2021; Sang-To et al., 2021; To, et al., 2022).

In this paper, a novel Swarm-inspired algorithm is proposed for solving optimization problems, named termite life cycle optimizer (TLCO), which is based on the termite life cycle and the modulation of movement strategies in nature. To the best of our knowledge, this is the first time TLCO is published in the literature.

2. Biological inspiration

Termites are social insects, widely distributed on Earth, which have reached a high level of social organization. The termite life cycle, shown in Fig. 2, is generally established by three groups of individuals: the worker caste, the soldier caste, and the reproductive caste (Keller, 1998). Thus, each caste will take on tasks to maintain the development of a colony. The life cycle is a typical of social insects allowing for proper division of labor. King and queen are only active reproductive individuals within a colony, and they perform no other functions. A queen can lay thousands of eggs each year. During the two-week incubation period, the termite worker takes care of the eggs. The nymphs hatch directly from the egg and can become one of three castes: the worker caste, the soldier caste, and the reproductive caste that are in charge of the following tasks:

Worker caste:

Termite workers represent 70 % to 80 % of the total number of insects in the colony. Workers undertake most of the work within the colony, being responsible for foraging, food storage, and brood and nest maintenance (Noirot & Pasteels, 1987).

Soldier caste:

The soldiers account for 20 % to 30 % of the number of insects within the colony. Their sole purpose is to protect the colony and attack intruders if they feel threatened (Thorne, Breisch, & Muscedere, 2003). To perform this task, they stay close to the nest and do not move too far from their colony.

Reproductive caste:

There is only one pair of reproductive individuals in a colony, a fertile female and male, known as the queen and king. The queen is responsible for egg production for the colony and the king mates with her for life. The queen starts producing new reproductive termites at a certain time of the year, and huge swarms emerge from the colony when nuptial flight begins. When they find a partner, these reproductive termites will lose their wings and create a new colony (Korb & Lenz, 2004).

Communication between termites is considered to be crucial element for the coordination of individuals' activities and the emergence of collective intelligence (Garnier, Gautrais, & Theraulaz, 2007). This characteristic is a significant factor for development and survival of the colony. Most termites are blind, so communication primarily occurs

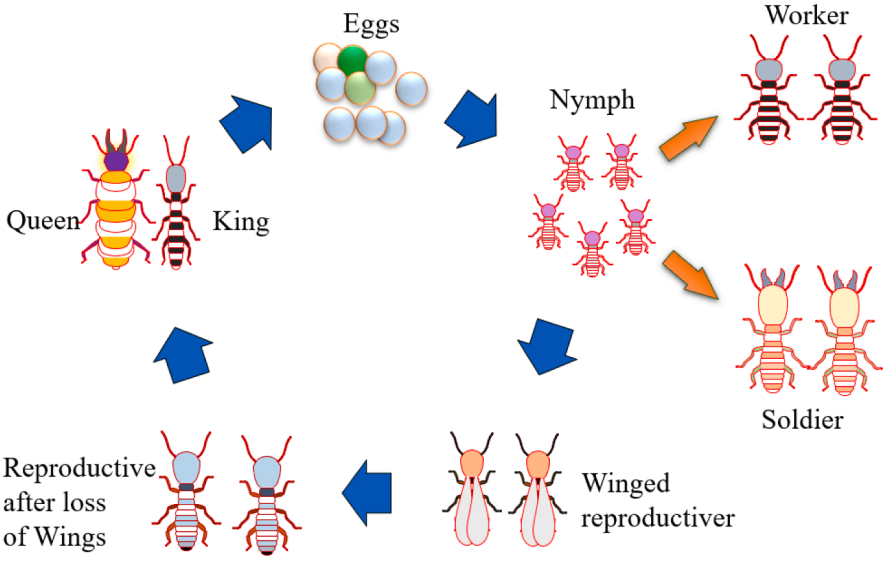

Fig. 2. Life cycle of a Termite colony.

Table 2
The conversion from the terms of termite life cycle to TLCO algorithm.

Terms of Termite life cycle	Task in colony	Task converted to TLCO
Queen	Lay eggs and take care of the Nymph	Global best solution
Eggs and Nymph	The source of development	The number of particles in swarm
Worker caste	Find the new food of source, build the shelter tubes	Particular particles that have the ability to explore the new search space
Soldier caste	Protect colony and attack the other intruders	Particular particles that have the ability to exploit the search space around the current global best
Reproductive caste	Find a new food source to create a new colony.	The ability to exploit new potential solution and to replace the termite workers and termite soldiers.
Intelligent swam	Communication of each termite in the colony	The ability to store information at the current iteration and transmit it to the next iteration.

through chemical signals and mechanical cues. Termites use this communication to share the location of food sources and organize the traffic outside the nest. The termite workers always leave pheromones on the ways to orient the others to the food.

TLCO takes inspiration from the specific tasks carried out by workers, soldiers and reproductive termites to build mathematical models that can reach three significant factors: (i) the ability of exploration and exploitation of the algorithm, (ii) the ability to share information among each particle in swarm and (iii) the ability to improve the solution over course of iterations. The conversion from the terms of termite life cycle to TLCO is described in Table 2.

3. Termite life cycle optimizer

This section provides the details of TLCO algorithm.

3.1. Random walk and Lévy flight

In nature, some movements performed by animals are mostly random (Benhamou, 2007). From a start point, the animal can move in any direction. For instance, such movements can be observed during foraging activity. In mathematical terms, a path can be represented by a succession of random steps as expressed in Eq. (1).

$$X(n) = \sum_{i=1}^{n} S_i = S_1 + S_2 + \ldots + S_n \tag{1}$$

where $S_i = (s_1, s_2, \ldots, s_D)$ denotes a step length. There have been many attempts to convert the step length to the mathematical formulation of probability. Many methods can determine these features, but the simplest one is the well-known Mantegna algorithm for asymmetric and stable Lévy distribution (Lévy & Lévy, 1954). If each step $S_i = (s_1, s_2, \ldots, s_D)$ in the random walk obeys a Lévy distribution, the random walk will become a Lévy flight (Yang & Deb, 2009). According to Mantegna's algorithm, the step length S_i is defined as in Eq. (2)

$$S = \frac{U}{|V|^{1/\beta}} \tag{2}$$

where β is the Lévy distribution index whose values are constrained as $1 \leqslant \beta \leqslant 2$.

U and V are drawn from normal distributions following Eq. (3)

$$U \tilde{\tilde{}} (0,\ \sigma_u^2), \quad V \tilde{\tilde{}} (0,\ \sigma_v^2) \tag{3}$$

where σ_u and σ_v are standard deviation given in Eq. (4)

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \sigma_v = 1 \tag{4}$$

In Eq. (4), the Gamma function Γ for an integer z is expressed as in Eq. (5)

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt \tag{5}$$

Each step length S can have both positive and negative values. Especially, a step length S_* can be achieved with either a long or a short distance depending on the parameter β as shown in Fig. 3a and Fig. 3b. Based on this feature, it is a robust method applied for the exploration of a large search space and the exploitation near the best solution in that space. These characteristics may provide some hints and insights into how and why metaheuristic algorithms behave.

3.2. A proposed modulation of step length in termite life cycle optimizer

The step length S in the original random walk is controlled by the parameter β whose values range from 1 to 2. Fig. 3a and Fig. 3b show that a long step length is decided by a small parameter value β and vice

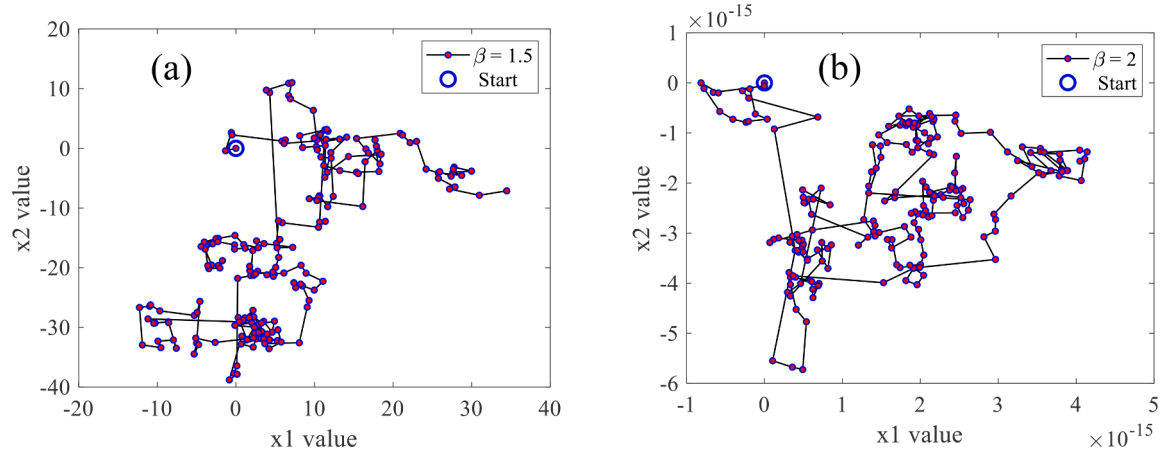

Fig. 3. Random walk in 2D dimension using Lévy flight in 200 steps length with different β: (a) simulation over long distance and (b) simulation over short distance.

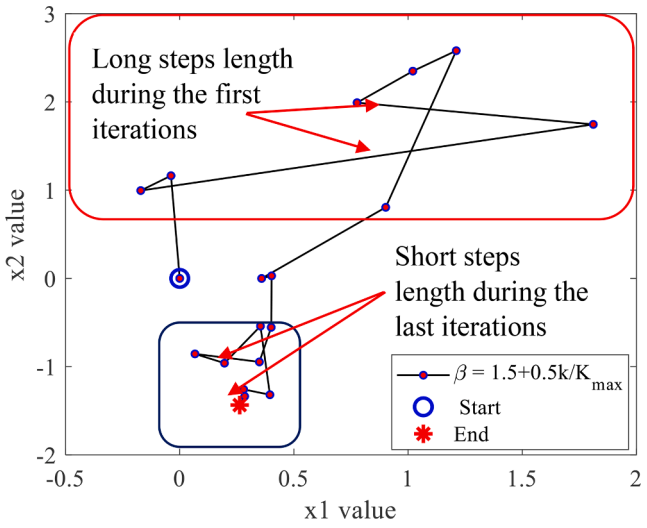

Fig. 4. Random walk in 2D using Lévy flight in 20 steps length with proposed parameter β.

versa. With a large parameter value β, a short step length is established. A successful optimization algorithm is to reach both conditions: (i) fast convergence rate and (ii) high accuracy level. These conditions require enough long movements to fast forward the best solution during the first iterations, and enough short movements to avoid the local optimal problem and to improve the accuracy level during the last iterations. In TLCO, this can be achieved by adjusting the value of parameter β at each iteration. As a result, the value of β is moving upwards over the course of iterations for improving convergence rate and accuracy level. To limit the search space during a first few iterations, the boundary condition of β value will be changed from range $[1,\ 2]$ to range $[1.5,\ 2]$ for improving convergence rate. Thus, the modification step length is presented in Eq. (6). The effect of the values of β on step length at each iteration is shown in Fig. 4.

$$\beta = 1.5 + \frac{0.5k}{K_{\max}}$$
$$S_*^{(k)} = \frac{U}{|V|^{1/\beta}} \tag{6}$$

where k is the current iteration and $K_{\max}$ is the maximum number of iterations.

3.3. The general movement strategy of termite life cycle optimizer

We assume that the position with high quality search space symbolized $\overrightarrow{X^*}$ is known. The movement strategy from the current position $\overrightarrow{X_{old}}$ to the new position $\overrightarrow{X_{new}}$ in 2D is normally given in Eq. (7).

$$\overrightarrow{X_{new}} = w_1\overrightarrow{X_{old}} + w_2\left(w_3\overrightarrow{X^*} - w_4\overrightarrow{X_{old}}\right) \tag{7}$$

where w_1,w_2, w_3 and w_4 are either a positive scalar vector or a scalar number having random distribution, which consider the influence of the orientation of each individual vector to the new position $\overrightarrow{X_{new}}$. Assuming that each vector $w_2,\ w_2,\ w_3,\ w_4$ is a scalar vector having its value within the range $[0,\ 1]$ and having a dimension $D = 2$, the process of movement using Eq. (7) is illustrated in Fig. 5a with 1st iteration. Furthermore, if we increase the number of iterations, we will reach the convergence to the value of $\overrightarrow{X^*}$ as shown in Fig. 5b.

It is obvious that, the search space using Eq. (7) is controlled by a set of vector $w = \{w_2,\ w_2,\ w_3,\ w_4\}$ and it is expanded from the position of $\overrightarrow{X_{old}}$ to $\overrightarrow{X^*}$ as shown in Fig. 5. The limiting of this search space causes

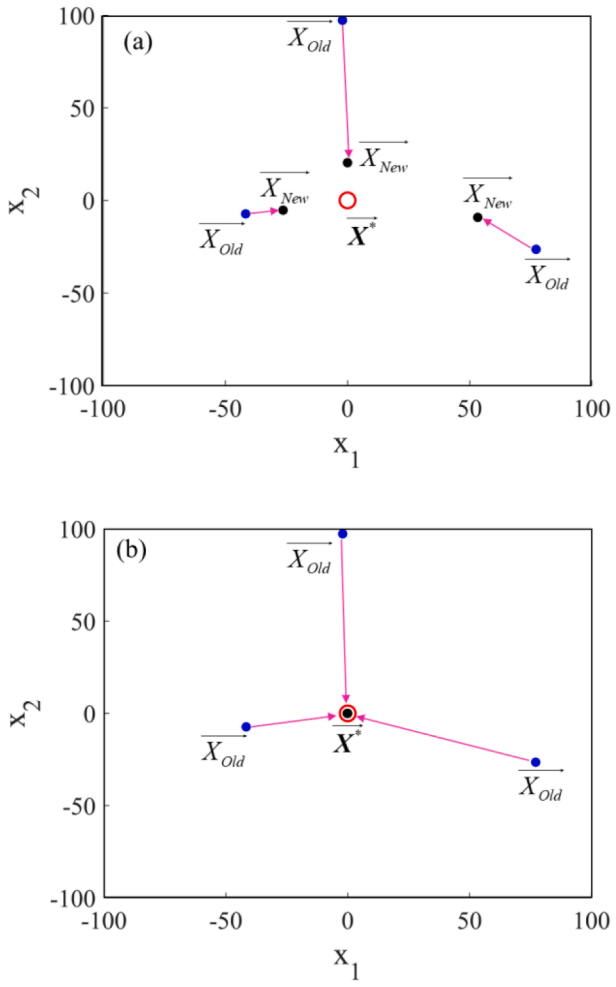

Fig. 5. The movement strategy using Eq. (7); (a) with 1st iteration, (b) with 10th iterations.

local optimization problems and it will face a big challenge when solving the problems with multiple local optima's. To overcome this limitation, a new technique is proposed to expand the search space based on the change of orientation movement. Thus, the vector $w_2\left(w_3\overrightarrow{X^*} - w_4\overrightarrow{X_{old}}\right)$ in Eq. (7) is replaced by its absolute value $\omega_2\left|\left(\omega_3\overrightarrow{X^*} - \omega_4\overrightarrow{X_{old}}\right)\right|$ at the same time, a new set of vector $\omega = \{\omega_1,\ \omega_2,\ \omega_3,\ \omega_4\}$ is also proposed to replace the old one $w = \{w_2,\ w_2,\ w_3,\ w_4\}$. The proposed movement strategy is given in Eq. (8)

$$\overrightarrow{X_{New}} = \omega_1\overrightarrow{X_{old}} + \omega_2\left|\omega_3\overrightarrow{X^*} - \omega_4\overrightarrow{X_{old}}\right| \tag{8}$$

where $\omega_1,\ \omega_2,\ \omega_3,\ \omega_4$ are either vectors or scalar numbers, which are used to control the expandable search space. These parameters will be flexibly selected to achieve both the new search space expansion and the convergence rate. Assuming that, the value of $\omega_1,\ \omega_3,\ \omega_4$ registers a dimension 2 and its value is random vector distributed in range $[0,\ 1]$, while the value of ω_2 is random vector having its value in range $[-1,\ 1]$, the movement strategy using Eq. (8) is shown in Fig. 6a and Fig. 6b with different iterations.

Both the movement strategies using Eq. (7) and Eq. (8) also led to the convergence to the value $\overrightarrow{X^*}$. However, it can be seen that the search space using Eq. (8) becomes more flexible and more diverse than using Eq. (7) as shown in Fig. 7. Based on this advantage, TLCO will make use of Eq. (8) to establish a general movement strategy, and the process of self-improvement of TLCO will be completed through the selection of a

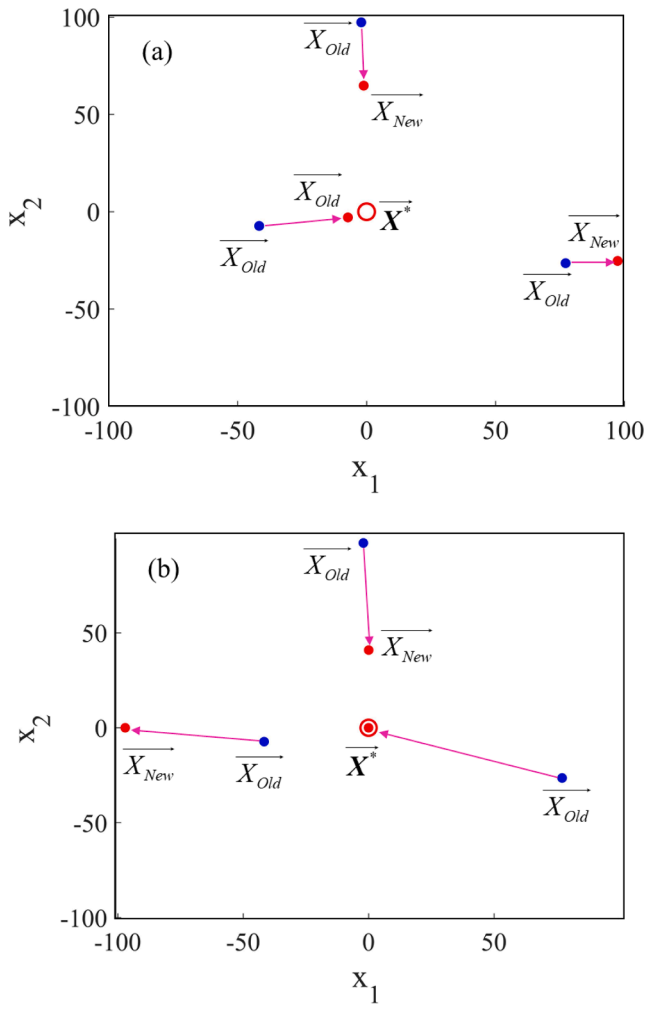

Fig. 6. The movement strategy using Eq. (8); (a) with 1st iteration, (b) with 10th iterations.

set of parameters ω_1, ω_2, ω_3, ω_4 to achieve a fast convergence rate and acceptable accuracy level based on the law of termite life cycle in nature.

3.4. The process of self-improvement in termite life cycle optimizer (TLCO)

Stochastic algorithm and population-based approach are used as the basis background for the development of TLCO, which starts with a set of initial population symbolized $\overrightarrow{P} = \left\{\overrightarrow{P}_1, \overrightarrow{P_2}, ..., \overrightarrow{P_N}\right\}$, where $\overrightarrow{P_i} = \{p_{i1}, p_{i2}, ..., p_{iD}\} \in \mathbb{R}^{1\times D}, (i = 1, 2, ..., N)$ defines a position in population (or an individual population) in dimension D. The self-improvement of TLCO is secured through the process of simulating how the population move forward to the search spaces having the high probability for exploiting the new global best values at each iteration. Thus, at each iteration, based on the law of termite life cycle in nature, the process of position updating of three castes; worker caste, soldier caste, and reproductive caste will be implemented to find the best solutions in each caste. Then, based on the Greedy selection strategy, the better solutions found will be compared to $\overrightarrow{P_i} = \{p_{i1}, p_{i2}, ..., p_{iD}\} \in \mathbb{R}^{1\times D}, (i = 1, 2, ..., N)$ for improving the initial population. If we reach a sufficient number of iterations, the convergence rate will be guaranteed. The process of self-improvement is shown Algorithm 1.

Algorithm 1: The process of self-improvement of initial population in TLCO over the course of iterations

For $k = 1{:}K_{max}$
For *each worker caste, the soldier caste, and the reproductive caste in colony*
Update the positions; $\overrightarrow{X_{i,worker}^{(k+1)}}$, $\overrightarrow{X_{i,soldier}^{(k+1)}}$ *and* $\overrightarrow{X_{i,reproductive}^{(k+1)}}$;
Calculate the fitness of all termites in colony
$f_{obj}\left(\overrightarrow{X_{i,worker}^{(k+1)}}\right)$, $f_{obj}\left(\overrightarrow{X_{i,soldier}^{(k+1)}}\right)$, $f_{obj}\left(\overrightarrow{X_{i,reproductive}^{(k+1)}}\right)$;
% Greedy selection to update P_i
$If f_{obj}\left(\overrightarrow{X_{i,worker}^{(k+1)}}\right) < f_{obj}\left(\overrightarrow{P_i^{(k)}}\right)$;
$\overrightarrow{P_i^{(k+1)}} = \overrightarrow{X_{i,worker}^{(k+1)}}, f_{obj}\left(\overrightarrow{P_i^{(k+1)}}\right) = f_{obj}\left(\overrightarrow{X_{i,worker}^{(k+1)}}\right)$
End
$If f_{obj}\left(\overrightarrow{X_{i,soldier}^{(k+1)}}\right) < f_{obj}\left(\overrightarrow{P_i^{(k)}}\right)$;
$\overrightarrow{P_i^{(k+1)}} = \overrightarrow{X_{i,soldier}^{(k+1)}}, f_{obj}\left(\overrightarrow{P_i^{(k+1)}}\right) = f_{obj}\left(\overrightarrow{X_{i,soldier}^{(k+1)}}\right)$
End
$If f_{obj}\left(\overrightarrow{X_{i,reproductive}^{(k+1)}}\right) < f_{obj}\left(\overrightarrow{P_i^{(k)}}\right)$;
$\overrightarrow{P_i^{(k+1)}} = \overrightarrow{X_{i,reproductive}^{(k+1)}}; f_{obj}\left(\overrightarrow{P_i^{(k+1)}}\right) = f_{obj}\left(\overrightarrow{X_{i,reproductive}^{(k+1)}}\right)$
End
End
End

The remaining problem to ensure the effectiveness of TLCO is to find a strategy for position updating of each caste in each iteration. And this process is described in detail in the remaining parts of this section.

3.4.1. The movement strategy of termite workers in model

In this paper, we assume that the number of termite workers accounts for 70 % of the total number of individuals in the colony. Let's assume an initial population of termites in the colony of size N; thus, $\overrightarrow{X_{i,worker}}$, $1 \leqslant i \leqslant 0.7N$ denotes the position of termite workers. The primary duties of termite workers are to explore the source of food and to build the shelter tubes. Based on this task, the movement strategy of them must start with their current position. And the orientation movement in worker caste must guarantee the ability of expanding the search space.

By converting $\overrightarrow{X^*}$ in Eq. (8) by $\overrightarrow{X_{best}^{(k)}}$, which is registered is the best solution at k^{th} iteration, the process of position updating between k^{th} and $(k+1)^{th}$ iteration can be described as in Eq. (9)

$$\overrightarrow{X_{i,worker}^{(k+1)}} = \omega_1 \overrightarrow{X_{i,worker}^{(k)}} + \omega_2 \left| \omega_3 \overrightarrow{X_{best}^{(k)}} - \omega_4 \overrightarrow{X_{i,worker}^{(k)}} \right| \tag{9}$$

To increase the velocity of the movement from $\overrightarrow{X_{i,worker}}$ to $\overrightarrow{X_{best}^{(k)}}$, the values of ω_1, ω_3 and ω_4 are selected equal one. Thus, Eq. (9) becomes Eq. (10).

$$\overrightarrow{X_{i,worker}^{(k+1)}} = \overrightarrow{X_{i,worker}^{(k)}} + \omega_2 \left| \overrightarrow{X_{best}^{(k)}} - \overrightarrow{X_{i,worker}^{(k)}} \right| \tag{10}$$

The ability of expanding the search space is determined by the parameter ω_2. In the previous section, a modification of step length is proposed based on the parameter β according to Eq. (6). Thus, the new step length will reach a long step length during the first iterations to expand the new search space and the short steps length during the last iterations to reduce the wasteful movements, which is denoted by $f_{ofun}(X_{i,worker}^{k}) > f_{ofun}(X_{i,worker}^{k+1})$, where f_{ofun} is the objective function. Based on the premise of convenience, the parameter ω_2 is presented in Eq. (11)

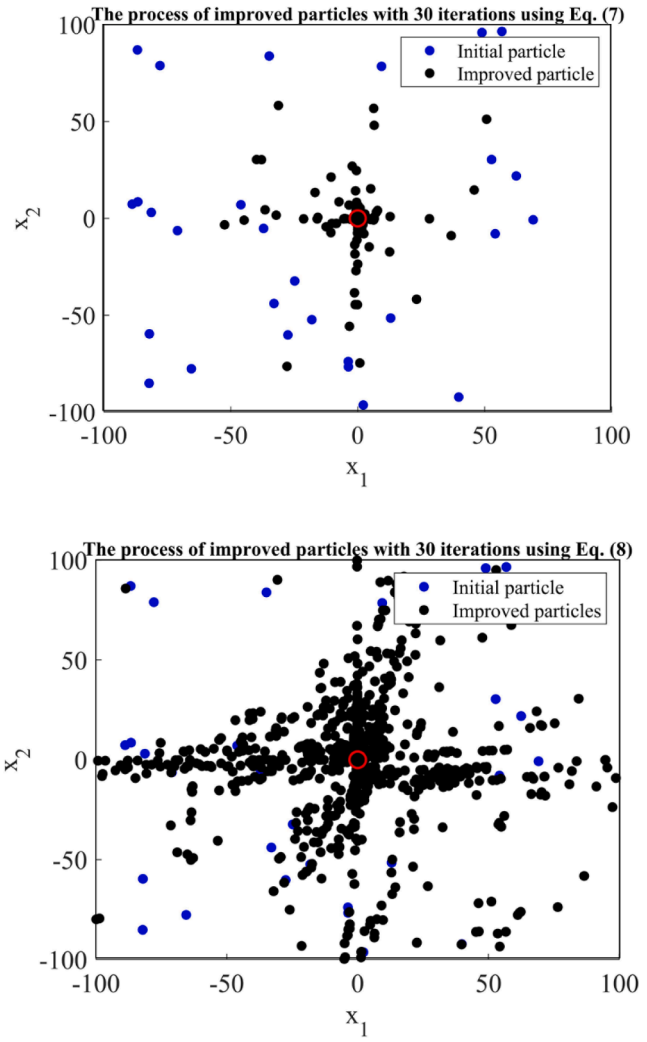

Fig. 7. The process of improved particles in population using Eq. (7) and Eq. (8).

$$\begin{aligned} \omega_2 &= \xi \times dS_*^{(k+1)} \\ dS_*^{(k+1)} &= \left(S_*^{(k+1)} + rand(1, D)\right) \end{aligned} \tag{11}$$

where ξ is a scalar number whose value is limited in range $[-1,\ 1]$, which is used to control the movement direction termite worker. Thus, at the step $(k+1)^{th}$, $\overrightarrow{X_{i,worker}^{(k+1)}}$ can randomly move in two directions, the first direction is secured if $\xi < 0$ and the remaining direction if $\xi > 0$.

$dS_*^{(k+1)}$ is a scalar vector to control the orientation movement of termite workers. It includes two terms; the first term is step length $S_*^{(k+1)}$ according to Eq. (6), while the second term is a scalar vector $rand(1, D)$ having dimension D, and its value is bounded within the range $[0,\ 1]$.

To simplify, Eq. (10) is rewritten as Eq. (12).

$$\overrightarrow{X_{i,worker}^{(k+1)}} = \overrightarrow{X_{i,worker}^{(k)}} + \xi \times \left(S_*^{(k+1)} + rand(1, D)\right) \otimes \left|\overrightarrow{X_{best}^{(k)}} - \overrightarrow{X_{i,worker}^{(k)}}\right| \tag{12}$$

where $\otimes$ is a point-to-point multiplication.

Because the vector $\left|\overrightarrow{X_{best}^{(k)}} - \overrightarrow{X_{i,worker}^{(k)}}\right|$ is the fixed-component. So, the ability of expand the search space depends on the component $\omega_2 = \xi \times dS_*^{(k+1)}$. The fluctuation of this vector is controlled by a vector $dS_*^{(k+1)}$, which gradually decreases as the number of iterations increases. Therefore, during the first few iterations, the amplitude of the fluctuation is large enough to expand the search space for improving the ability of exploration, and the amplitude will be steadily declined as the number of iterations increases as shown in Fig. 8. This will lead to the search space getting smaller during the last few iterations to avoid the waste movement.

3.4.2. The movement strategy of termite soldiers in the model

The number of termite soldiers represents about 30 % of the total number of individuals within a colony. Let's $\overrightarrow{X_{i,soldier}},\ 0.7N < i \leqslant N$ be the position of a termite soldier, the primary task of this caste is to protect the colony and attack the intruders. To complete the mission, the movement of termite soldiers remains close to their colony to protect the queen termite called $\overrightarrow{X_{best}}$. Based on this task, the movement strategy of them must start with $\overrightarrow{X_{best}}$.

By replacing $\overrightarrow{X_{old}}$ by $\overrightarrow{X_{best}^{(k)}}$ and $\overrightarrow{X^*}$ by $\overrightarrow{X_{i,soldier}}$, the process of position updating between k^{th} and $(k+1)^{th}$ iteration can be described as in Eq. (13)

$$\overrightarrow{X_{i,soldier}^{(k+1)}} = \overrightarrow{\omega_1 X_{best}^{(k)}} + \omega_2 \left|\omega_3 \overrightarrow{X_{i,soldier}^{(k)}} - \omega_4 \overrightarrow{X_{best}^{(k)}}\right| \tag{13}$$

The parameter ω_1 is selected as a scalar number and its value ranges in the interval $[0,\ 2]$ to consider the gap between the position of termite soldiers and the colony. The parameter ω_2 is a scalar number having value within the interval $[-1,\ 1]$ as the same as ξ in Eq. (12). Thus, termite soldiers register two movement direction depending the value of $\omega_2 > 0$ or $\omega_2 < 0$.

During the last few iterations, through the process of self-improvement, $\overrightarrow{X_{i,soldier}}$ will be closer to $\overrightarrow{X_{best}}$. If the vector $\left|\omega_3 \overrightarrow{X_{i,soldier}^{(k)}} - \omega_4 \overrightarrow{X_{best}^{(k)}}\right|$ is not to reach desired distance, it leads to the probability of getting stuck at local optimal as shown in Fig. 9. To get around this limitation, the parameter ω_3 is selected equal to one and ω_4 is assigned by a modification of step length as in Eq. (6).

To simplify, Eq. (13) is rewritten as Eq. (14).

$$\begin{aligned} \overrightarrow{X_{i,soldier}^{(k+1)}} = 2 \times rand(0,1) \times \overrightarrow{X_{best}^{(k)}} + (-1 + 2 \times rand(0,1)) \times \Big|\overrightarrow{X_{i,soldier}^{(k)}} \\ - S_*^{(k+1)} \otimes \overrightarrow{X_{best}^{(k)}}\Big| \end{aligned} \tag{14}$$

Because of the increase of the parameter β, the vector of $S_*^{(k+1)}$ gradually decreases as the number of iterations increases and $S_*^{(k+1)}$ will reach approximately zero at the last iterations. Thus, the distance of $\left|\omega_3 \overrightarrow{X_{i,soldier}^{(k)}} - \omega_4 \overrightarrow{X_{best}^{(k)}}\right|$ is long enough to limit the local optimal problem

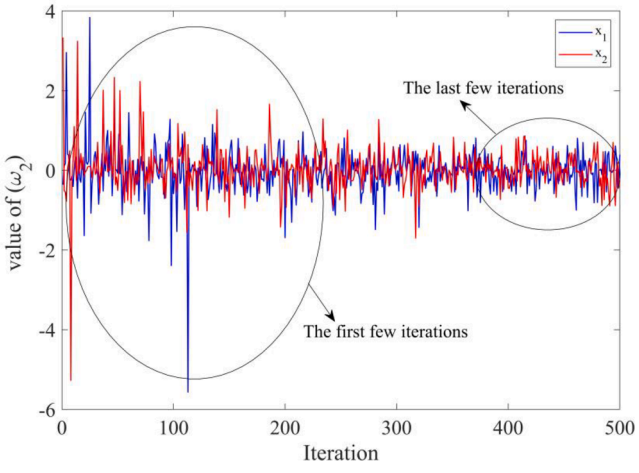

Fig. 8. The distribution of (ω_2) with 500 iterations.

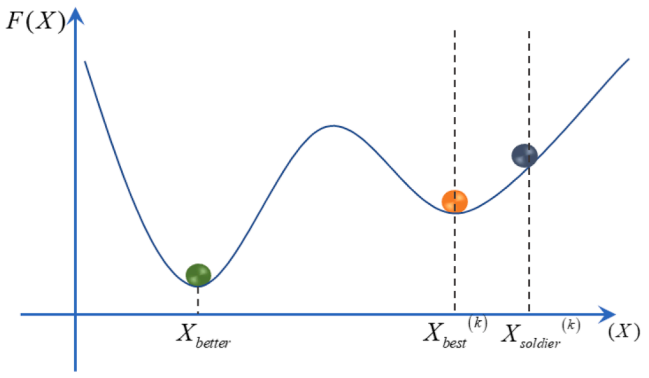

Fig. 9. The ability of getting stuck at local optimal during the last few iterations.

during the last iterations. With the appropriate selection of a set of parameters ω_1, ω_2, ω_3 and ω_4, the movement of termite soldier is correctly simulated its actual task. In mathematical model, this strategy will find the potential search space around $\overrightarrow{X_{best}}$, at the same time, it is also not too close to $\overrightarrow{X_{best}}$ to overcome the local optimal (Fig. 10).

3.4.3. The movement strategy of reproductive termites in the model

The primary task of reproductive termites is to create a new colony and replace the termite workers and termite soldiers. In a mathematical model, this process can be achieved by evaluating the performance of each termite worker $\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}}$ and each termite soldier $\overrightarrow{\mathrm{X}_{\mathrm{i,soldier}}}$ over the course of iterations. If $\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}}$ and $\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}}$ perform wasteful movements too frequently, the reproductive termites $\overrightarrow{\mathrm{X}_{\mathrm{i,reproductive}}}$ will emerge to find the new potential search space and replace the poor search space found by $\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}}$ and $\overrightarrow{\mathrm{X}_{\mathrm{i,soldier}}}$. The appearance of reproductive termites aims to increase the opportunity for escaping the local optimization problem and of finding a better search spaces. Thus, after each iteration if each $\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}}$ and $\overrightarrow{\mathrm{X}_{\mathrm{i,soldier}}}$ cannot explore a new better food source; in other words, if the condition $f_{ofun}(\overrightarrow{X_{i,worker}^{(k)}}) < f_{ofun}(\overrightarrow{P_i^{(k)}})$ and $f_{ofun}(\overrightarrow{X_{i,soldier}^{(k)}}) < f_{ofun}(\overrightarrow{P_i^{(k)}})$ as mention in **Algorithm 1** are not satisfied, these times will be recorded and counted through iterations by a pre-determined number called $Trial_{i,worker}$ and $Trial_{i,soldier}$ following **Algorithm 2.**

Algorithm 2: The process of recording the wasteful movements of each $\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}}$ and $\overrightarrow{\mathrm{X}_{\mathrm{i,soldier}}}$ over the course of iterations

For *each termite worker* $\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}}$ and $\overrightarrow{\mathrm{X}_{\mathrm{i,soldier}}}$ *in colony*
IF $f_{ofun}(\overrightarrow{X_{i,worker}^{(k+1)}}) > f_{ofun}(\overrightarrow{P_i^{(k+1)}})$;
$Trial_{i,worker}^{(k+1)} = Trial_{i,worker}^{(k)} + 1$;
End
IF $f_{ofun}(\overrightarrow{X_{i,soldier}^{(k+1)}}) > f_{ofun}(\overrightarrow{P_i^{(k+1)}})$;
$Trial_{i,soldier}^{(k+1)} = Trial_{i,soldier}^{(k)} + 1$;
End
End

TLCO proposed two functions control the times of reproductive termites' emergence for replacement $\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}}$ and $\overrightarrow{\mathrm{X}_{\mathrm{i,soldier}}}$ at t^{th} iteration given in Eq. (15) and Eq. (16). And the shape of functions is shown in Fig. 11.

$$\lambda_{worker}(k) = 1 - \frac{1}{1 + e^{-\sigma(k - 0.5K_{\max})}} \tag{15}$$

$$\lambda_{soldier}(k) = \frac{1}{1 + e^{-\sigma(k - 0.5K_{\max})}} \tag{16}$$

where σ is constant number given in Eq. (17)

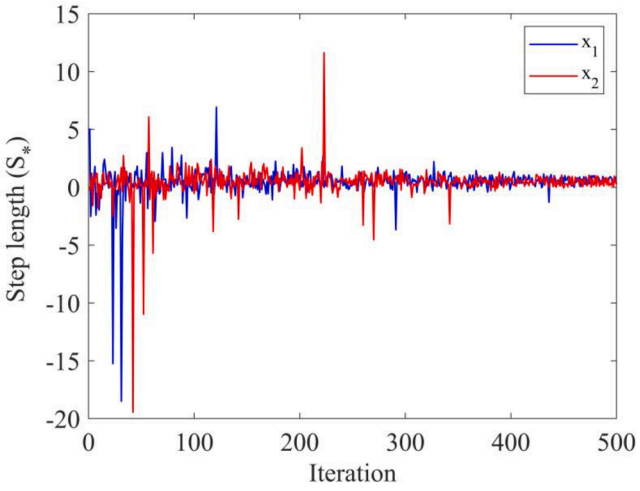

Fig. 10. The distribution of (S_*) with 500 iterations.

$$\sigma = \frac{1}{0.1K_{\max}} \tag{17}$$

where K_{max} is the maximum number off iterations.

Based on two proposed functions, the reproductive termites $\overrightarrow{\mathrm{X}_{\mathrm{i,reproductive}}}$ will appear to fly to a new potential region for establishing a new colony and replacing $\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}}$ and $\overrightarrow{\mathrm{X}_{\mathrm{i,soldier}}}$ following two conditions given in Eq. (18) and Eq. (19).

$$If \left(\frac{Trial_{i,worker}^{(k)}}{K_{\max}}\right) \geqslant \lambda_{worker}(k) \tag{18}$$

$$If \left(\frac{Trial_{i,soldier}^{(k)}}{K_{\max}}\right) \geqslant \lambda_{soldier}(k) \tag{19}$$

Because the initial population $\overrightarrow{P} = \left\{\overrightarrow{P_1}, \overrightarrow{P}_2, ..., \overrightarrow{P_N}\right\}$ will be improved after each iteration by the greedy selection strategy following **Algorithm 1**, the search space between each individual population $\overrightarrow{P_i} = \{p_{i1}, p_{i2}, ..., p_{iD}\} \in \mathbb{R}^{1\times D}, (i = 1, 2, ..., N)$ is the destination of reproductive termites given in Eq. (20)

$$\begin{aligned}\overrightarrow{X_{i,reproductive}^{(k+1)}} &= \overrightarrow{P_{r1}^{(k)}} + rand(1,D) \otimes \left(\overrightarrow{P_{r2}^{(k)}} - \overrightarrow{P_{r1}^{(k)}}\right) \quad If\ f_{ofun}\left(\overrightarrow{P_{r2}^{(k)}}\right) < f_{ofun}\left(\overrightarrow{P_{r1}^{(k)}}\right) \\ \overrightarrow{X_{i,reproductive}^{(k+1)}} &= \overrightarrow{P_{r1}^{(k)}} + rand(1,D) \otimes \left(\overrightarrow{P_{r1}^{(k)}} - \overrightarrow{P_{r2}^{(k)}}\right) \quad else\end{aligned} \tag{20}$$

where indexes r_1, $r_2 \in (1, 2, ..., N)$, are random selected from running index $i = (1, 2, ..., N)$ and r_1 is different fromr_2. In some cases, the global optima's will be located in a region, that is too far away from the position of $\overrightarrow{P_{r1}}$ and $\overrightarrow{P_{r2}}$. In other words if Eq. (20) is unable to reach the ability of expanding the large search space, TLCO will face a risk of local optimal problem. To overcome this limitation, the search spaces will be renewed following Eq. (21)

$$\begin{aligned}\overrightarrow{X_{i,reproductive}^{(k+1)}} = \overrightarrow{L_b} + rand(1,D)\Big(\overrightarrow{U_b} \\ - \overrightarrow{L_b}\Big) \quad If\ f_{ofun}\left(\overrightarrow{X_{i,reproductive}^{(k+1)}}\right) < f_{ofun}\left(\overrightarrow{P_i^{(k)}}\right)\end{aligned} \tag{21}$$

where $\overrightarrow{L_b}$ and $\overrightarrow{U_b}$ are the lower-bound and upper-bound vectors, respectively, and $\overrightarrow{D}$ is the dimension of the objective function.

It can be seen that two proposed functions Eq. (15) and Eq. (16)

ensure consistency with the movement strategy of $\overrightarrow{X_{i,worker}}$ and $\overrightarrow{X_{i,soldier}}$, respectively. In TLCO, the ability of exploration is bias to $\overrightarrow{X_{i,worker}}$ and the ability of exploitation is bias to $\overrightarrow{X_{i,soldier}}$. Thus, based on the distribution values of $\lambda_{worker}(t)$ and $\lambda_{soldier}(t)$, during the first few iterations, the probability of Eq. (19) is greater than the probability of Eq. (18). So, the emergence of reproductive termites $\overrightarrow{X_{i,reproductive}}$ to replace soldier termites $\overrightarrow{X_{i,soldier}}$ will be more biased than worker termites $\overrightarrow{X_{i,worker}}$. It leads to the less fluctuation of worker termites $\overrightarrow{X_{i,worker}}$ and allows them to move faster to high-quality search space following Eq. (12) for improving the convergence rate. Meanwhile, during the last few iterations, the probability of Eq. (18) is greater than the probability of Eq. (19). It leads to the less fluctuation of soldier termites $\overrightarrow{X_{i,soldier}}$ and allows them to move around to the best solution given in Eq. (14) for improving the accuracy level. At the same time, the fluctuation of $\overrightarrow{X_{i,worker}}$ will improve the ability of expanding the search space for escaping the local optimal problem. The typical fluctuation of $\overrightarrow{X_{i,worker}}$ and $\overrightarrow{X_{i,soldier}}$ based on the distribution of $\lambda_{worker}(t)$ and $\lambda_{soldier}(t)$ over the course iterations is illustrated in Fig. 12.

Based on the above assessment and analysis, we make sure that TLCO guarantees to achieve the robust balance between the ability of exploration and exploitation over the course of iterations. The emergent process of reproductive termites is given in **Algorithm 3**.

Algorithm 3: The emergent process of reproductive termites over the course of iterations

For $k = 1{:}K_{max}$
For *each termite worker in colony*
IF $\left(\frac{Trial_{i,worker}^{(k+1)}}{K_{\max}}\right) \geqslant \lambda_{worker}(k+1)$;
$Trial_{i,worker}^{(k+1)} = 0$;
$$\overrightarrow{X_{i,reproductive}^{(k+1)}} = \overrightarrow{P_{r1}^{(k)}} + rand(1,D) \otimes \left(\overrightarrow{P_{r2}^{(k)}} - \overrightarrow{P_{r1}^{(k)}}\right) \quad If\ f_{ofun}\left(\overrightarrow{P_{r2}^{(k)}}\right) < f_{ofun}\left(\overrightarrow{P_{r1}^{(k)}}\right)$$
$$\overrightarrow{X_{i,reproductive}^{(k+1)}} = \overrightarrow{P_{r1}^{(k)}} + rand(1,D) \otimes \left(\overrightarrow{P_{r1}^{(k)}} - \overrightarrow{P_{r2}^{(k)}}\right) \quad else \quad ;$$
Calculate the fitness;$f_{obj}\left(\overrightarrow{X_{i,reproductive}^{k+1}}\right)$
IF $f_{obj}\left(\overrightarrow{X_{i,reproductive}^{(k+1)}}\right) < f_{obj}\left(\overrightarrow{P_i^{(k)}}\right)$;
$\overrightarrow{P_i} = \overrightarrow{X_{i,reproductive}^{(k+1)}}, f_{obj}\left(\overrightarrow{P_i^{k}}\right) = f_{obj}\left(\overrightarrow{X_{i,reproductive}^{(k+1)}}\right)$
Update $\overrightarrow{X_{best}}$ *and the best objective function* $f\left(\overrightarrow{X_{best}}\right)$;
Else
$\overrightarrow{X_{i,reproductive}^{(k+1)}} = L_b + rand(1,D) \otimes (U_b - L_b)$; *where* (U_b, L_b) *are the boundary conditions*
End
$\overrightarrow{X_{i,worker}^{(k+1)}} = \overrightarrow{X_{i,reproductive}^{(k+1)}}$;
End
End
%——————————————————————————————

For each *termite soldier in colony*
IF $\left(\frac{Trial_{i,soldier}^{(k+1)}}{K_{\max}}\right) \geqslant \lambda_{soldier}(k+1)$;
$Trial_{i,soldier}^{(k+1)} = 0$;
$$\overrightarrow{X_{i,reproductive}^{(k+1)}} = \overrightarrow{P_{r1}^{(k)}} + rand(1,D) \otimes \left(\overrightarrow{P_{r2}^{(k)}} - \overrightarrow{P_{r1}^{(k)}}\right) \quad If\ f_{ofun}\left(\overrightarrow{P_{r2}^{(k)}}\right) < f_{ofun}\left(\overrightarrow{P_{r1}^{(k)}}\right)$$
$$\overrightarrow{X_{i,reproductive}^{(k+1)}} = \overrightarrow{P_{r1}^{(k)}} + rand(1,D) \otimes \left(\overrightarrow{P_{r1}^{(k)}} - \overrightarrow{P_{r2}^{(k)}}\right) \quad else \quad ;$$
Calculate the fitness;$f_{obj}\left(\overrightarrow{X_{i,reproductive}^{(k+1)}}\right)$;
IF $f_{obj}\left(\overrightarrow{X_{i,reproductive}^{(k+1)}}\right) < f_{obj}\left(\overrightarrow{P_i^{(k)}}\right)$;
$\overrightarrow{P_i^{(k+1)}} = \overrightarrow{X_{i,reproductive}^{(k+1)}}; f_{obj}\left(\overrightarrow{P_i^{k}}\right) = f_{obj}\left(\overrightarrow{X_{i,reproductive}^{(k+1)}}\right)$
Update $\overrightarrow{X_{best}}$ *and the best objective function* $f\left(\overrightarrow{X_{best}}\right)$;
Else
$\overrightarrow{X_{i,reproductive}^{(k+1)}} = \overrightarrow{L_b} + rand(1,D) \otimes \left(\overrightarrow{U_b} - \overrightarrow{L_b}\right)$; *where* $\left(\overrightarrow{U_b}, \overrightarrow{L_b}\right)$ *are the boundary conditions*
End
$\overrightarrow{X_{i,soldier}^{(k+1)}} = \overrightarrow{X_{i,reproductive}^{(k+1)}}$;
End
End

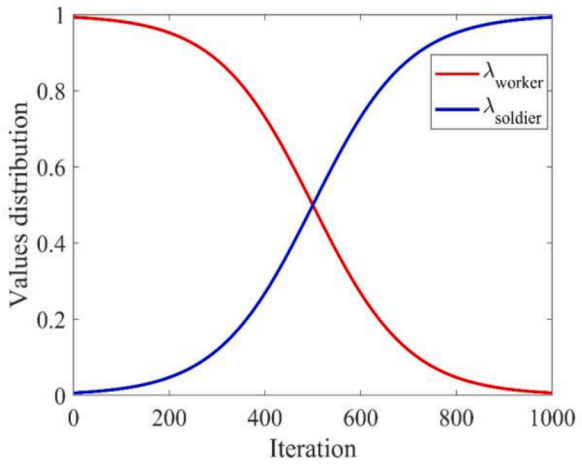

Fig. 11. The functions control the times of reproductive termites' emergence in TLCO with 1000 iterations.

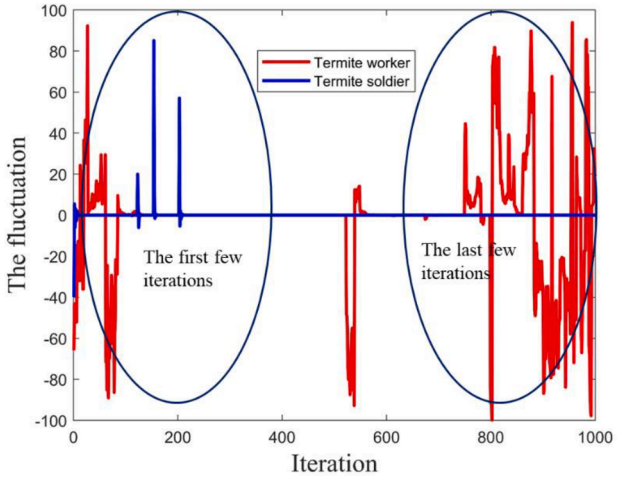

Fig. 12. The fluctuation shape of termite worker and soldier based on two proposed functions $\lambda_{worker}(t)$ and $\lambda_{soldier}(t)$ with 1000 iterations.

3.5. Schematic representation of termite life cycle optimizer (TLCO)

According to the descriptions of TLCO in the previous sections, some primary characteristics can be summarized to show how TLCO can be effective for solving optimization problems:

- It is not like the other algorithms, the search process will focus on the self-improvement of each agent search, which is sometimes cause local optimization problems. In TLCO, the search process is secured through the self-improvement of an initial population $\left(\overrightarrow{P_i}\right)$, $i = 1, 2, ..., N$ according to **Algorithm 1**. By this way, the adjustment direction of the movement of each termite workers, soldiers and reproductive termites becomes more flexible because the information gained from them at any iteration is stored by its compatible individual position $P_i = \{p_{i1}, p_{i2}, ..., p_{iD}\} \in \mathbb{R}^{1\times D}, (i = 1, 2, ..., N)$ in the initial population.
- TLCO includes a modulation of step length S_*, which is controlled by parameter β whose value increases from 1.5 to 2 over the course of iterations. This ensures that TLCO covers: (1) long distance during a first few iterations so as to improve the convergence rate and (2) enough short distance during the a last few iterations to enhance the level of accuracy and avoid the wasteful movement.
- TLCO has the ability to exploit and explore the new search space in a complete way in comparison with other optimization algorithms inpired by the behaviors of biological collectives shown in Table 1. For the first time, in accordance with the relative proportion of termite workers and soldiers found in a colony, TLCO includes a number of particles whose 70 % is termite workers and 30 % is termite soldiers. Thus, termite workers are biased to the ability of exploration and termite soldiers are biased to the ability of exploitation.
- The space of exploration or exploitation is controlled by the value of the step length S_* and two parameters ξ and ω_2 whose values are in the range $[-1, 1]$. Based on these parameters, the movement direction in TLCO becomes more flexible to expand the search spaces.
- The ability of exploration the new search space is a great advantage of TLCO compared to other algorithms. Based on two proposed functions, the appearance condition of reproductive termites is more bias to the ability of exploration during the first few iterations. During the last few iterations, a new set of search space is established to reset the movement direction of termite workers. This leads to the appreciative ability of escaping local optimal problem.

Based on the combination of **Algorithm 1**, **Algorithm 2** and **Algorithm 3**, the movement strategy of termite workers, termite soldiers and reproductive termites, the pseudo code of TLCO is given in **Algorithm 4.**

Table 3
Description of uni-modal benchmark functions.

Function	Solution space (S)	$f_{\min}$
$F_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^D$	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-100, 100]^D$	0
$F_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-100, 100]^D$	0
$F_4(x) = \max\{\lvert x_i \rvert, 1 \leqslant i \leqslant n\}$	$[-100, 100]^D$	0
$F_5(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right) + (x_i - 1)^2\right]$	$[-30, 30]^D$	0
$F_6(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	$[-100, 100]^D$	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + rand(0, 1)$	$[-1.28, 1.28]^D$	0

Algorithm 4: The implementation process of TLCO to find the best solution

%% Start

Select the initial population size(N), the maximum iterations($K_{\max}$);

Initialize the termite population $\overrightarrow{P_i}$ $(i = 1, 2, \ldots, N)$;

Calculate the objective function of each individual population $f_{obj}\left(\overrightarrow{P_i}\right)$, $i = 1, 2, \ldots, N$;

Find the best solution $\overrightarrow{X_{best}}$ *and the best objective function* $f\left(\overrightarrow{X_{best}}\right)$;

*%% **Main loop***

For $k = 1$: K_{max}

Calculate the modification of step length S_* *at each iteration using Eq.* ***Error! Reference source not found.Error! Reference source not found.****;*

% Start the tasks of termite workers

For *each termite worker* $\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}}$, $(1 \leqslant i < 0.7N)$

Update the position of each termite worker using Eq. (12);

Calculate the objective function of each termite work $f_{obj}\left(\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}^{(k+1)}}\right)$;

IF $\mathrm{f}_{obj}\left(\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}^{(k+1)}}\right) < \mathrm{f}_{obj}\left(\overrightarrow{P_i^{(k)}}\right)$

$\overrightarrow{P_i^{(k)}} = \overrightarrow{\mathrm{X}_{\mathrm{i,worker}}^{(k+1)}}, \mathrm{f}_{obj}\left(\overrightarrow{\mathrm{X}_{\mathrm{i,worker}}^{(k+1)}}\right) = \mathrm{f}_{obj}\left(\overrightarrow{P_i^{(k)}}\right)$;

Update $\overrightarrow{X_{best}}$ *and the best objective function* $f\left(\overrightarrow{X_{best}}\right)$;

Else

$Trial_{i,worker}^{(k+1)} = Trial_{i,worker}^{(k)} + 1$;

End

% Start the tasks reproductive termites

IF $\left(\frac{Trial_{i,worker}^{(k+1)}}{K_{\max}}\right) \geqslant \lambda_{worker}(k+1)$

Apply the ***Algorithm 3*** *incase termite workers;*

End

End

% Stop the tasks of reproductive termites

% Stop the tasks of termite workers

% Start the tasks of termite soldiers

For *each termite soldier* $\overrightarrow{\mathrm{X}_{\mathrm{i,soldier}}}$, $(0.7N < i <= N)$

Update the position of each termite worker using Eq. (14);

Calculate the objective function of each termite work $f_{obj}\left(\overrightarrow{\mathrm{X}_{\mathrm{i,soldier}}^{(k+1)}}\right)$;

IF $\mathrm{f}_{obj}\left(\overrightarrow{\mathrm{X}_{\mathrm{i,soldier}}^{(k+1)}}\right) < \mathrm{f}_{obj}\left(\overrightarrow{P_i^{(k)}}\right)$

$\overrightarrow{P_i^{(k+1)}} = \overrightarrow{\mathrm{X}_{\mathrm{i,soldier}}^{(k+1)}}, \mathrm{f}_{obj}\left(\overrightarrow{\mathrm{X}_{\mathrm{i,soldier}}^{(k+1)}}\right) = \mathrm{f}_{obj}\left(\overrightarrow{P_i^{(k)}}\right)$;

Update $\overrightarrow{X_{best}}$ *and the best objective function* $f\left(\overrightarrow{X_{best}}\right)$;

Else

$Trial_{i,soldier}^{(k+1)} = Trial_{i,soldier}^{(k)} + 1$;

End

% Start the tasks reproductive termites

IF $\left(\frac{Trial_{i,soldier}^{(k+1)}}{K_{\max}}\right) \geqslant \lambda_{soldier}(k+1)$;

Apply the ***Algorithm 3*** *incase termite soldiers;*

End

End

% Stop the tasks of reproductive termites

% Stop the tasks of termite workers

%

Out put the best result at each iteration

End

4. Numerical examples

4.1. Classical benchmark functions

To demonstrate the effectiveness and reliability of TLCO in solving optimization problems, we have tested 23 classical benchmark functions investigated in previous studies (Le-Duc, Nguyen, & Nguyen-Xuan, 2020; Mirjalili, 2016b; Mirjalili, et al., 2014). The first family test functions (F1-F7) shown in Table 3 have only one global optimum with no local optima. These functions are employed to test the abilities of convergence rate and exploitation. The second group (F8-F13) shown in Table 4 has multiple local solutions in addition to a global optimum. These functions are employed to test the ability of the algorithm to escape from local optima and explore the new search space. The final group gathers fixed-dimensional multi-modal functions (F14-F23) shown in Table 5. Note that the difference between the groups (F8-F13)

Table 4
Description of multi modal benchmark functions.

Function	Solution space (S)	$f_{\min}$
$F_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]^D$	$-418.9829 \times D$
$F_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-5.15, 5.12]^D$	0
$F_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^D$	0
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-500, 500]^D$	0
$F_{12}(x) = \frac{\pi}{x}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2 + \sum_{i=1}^{n} u(x_i, 10, 100, 4)\right\}$, $y_i = 1 + \frac{x_i + 1}{4}$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]^D$	0
$F_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]^D$	0

Table 5
Description of fixed-dimension multi-modal benchmark functions.

Function	Solution space (S)	$f_{\min}$
$F_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	$[-65.53, 65.53]^D$	1
$F_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1\left(b_i^2 + b_i x_2\right)}{b_i^2 + b_i x_3 + x_4}\right]^2$	$[-5, 5]^D$	0.0003
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]^D$	−1.0316
$F_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	$[-5, 10] \times [0, 15]$	0.398
$F_{18}(x) = \left(1 + (x_1 + x_2 + 1)^2\right)\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right) \times \left(30 + (2x_1 - 3x_2)^2\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right)$	$[-5, 5]^D$	3
$F_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	$[0, 1]^D$	−3.86
$F_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	$[0, 1]^D$	−3.32
$F_{21}(x) = -\sum_{i=1}^{5}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	$[0, 10]^D$	−10.1532
$F_{22}(x) = -\sum_{i=1}^{7}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	$[0, 10]^D$	−10.4028
$F_{23}(x) = -\sum_{i=1}^{10}\left[(X - a_i)(X - a_i)^T + c_i\right]^{-1}$	$[0, 10]^D$	−10.5363

and (F14-F23) lies in the ability to define the desired number of design variables. The characteristic of fixed-dimensional test functions is not to allow changes in the number of design variables, but they provide different search space in comparison with multi-modal test functions (F8-F13).

The performance of TLCO will be evaluated through different metrics. Six metrics are used to describe the performance of TLCO including convergence rate of the objective functions, search history of whole termite workers, termite soldiers and reproductive termites in 2*D*. Fig. 13 show some typical functions including the uni-modal functions (F1, F4, F6) and the multimodality (F8, F14, F23). TLCO uses a number of termites' $N = 30$ during 100 iterations to improve the initial population. The number of termites is divided in two groups, named the workers the soldiers: $\left(X_{i,worker},\ i = 1, 2, ..., 0.7N\right)$ and $\left(X_{i,soldier},\ i = 0.7N, ..., N\right)$, respectively.

4.2. The performance of TLCO

The balance between exploration and exploitation is the primary factor for a successful algorithm. TLCO clearly assign these skills to termite soldiers and termite workers in the colony, respectively. As the same time, reproductive termites will occur when the termite worker or termite soldier have difficulties to find a new food source to establish a new colony. The appearance of the reproductive termite will greatly improve the ability of exploration of TLCO because it creates a new set of search spaces to replace the old ones, which is not appreciated.

The search spaces, which are recorded by whole termite workers, termite soldiers and reproductive termites over 100 iterations, are shown in Fig. 13 (the second column, third column at the first row and the first column at the second row). It can be noticed that the movement strategy of termite workers will cover a wide search space during the early iterations based on the fluctuation of modification of step lengths S_* This trend is more evident in almost all functions characterized by both un-imodal (F1, F4, F6) and multimodality (F8, F14, F23), when the density of the termite workers is almost spread across the whole search space in boundary condition. Based on this wide distribution, TLCO reaches many opportunities to find the new best solutions for improving the convergence rate and escape from local optima. As the number of iterations increases, the search spaces are shrinking gradually because of

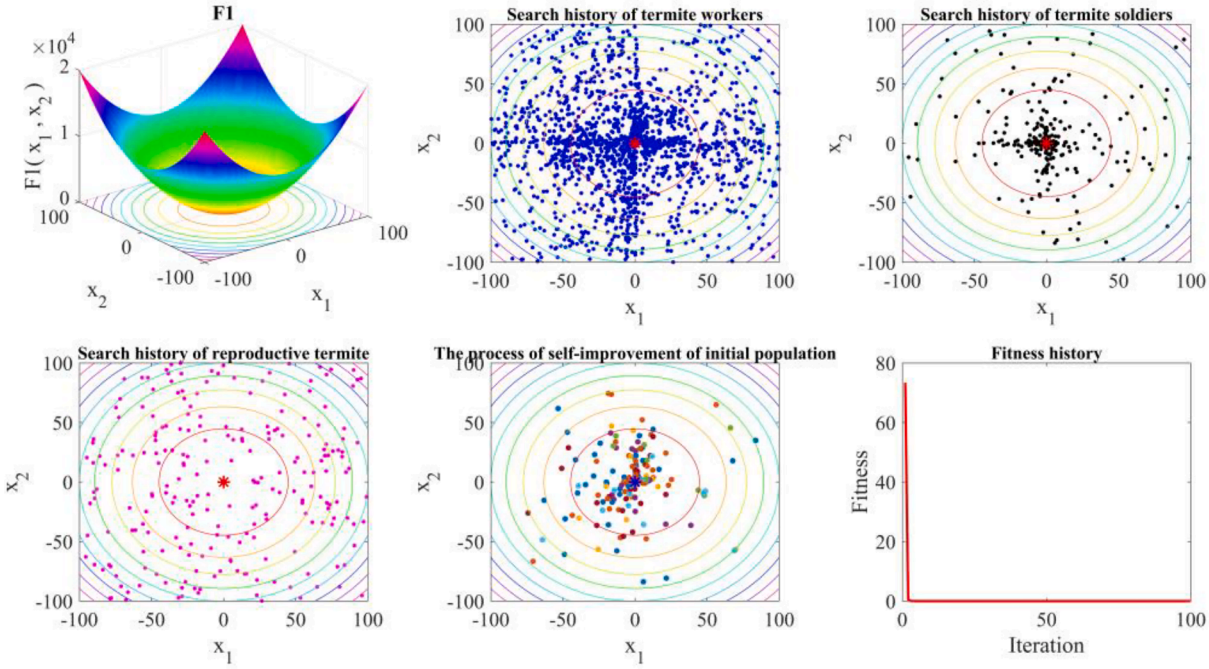

Fig. 13. Quantitative results of TLCO for three groups of typical functions.

the control of the step lengthS_*. As a consequence, the search space will be getting smaller, and the density of termite workers will only be gathered around the search space, which is the highest quality as shown in the second column at the first row in Fig. 13. The ability of exploitation in the TLCO is guaranteed by the termite soldiers as shown in the third column at the first row in Fig. 13. Note that the processes of exploitation and exploration in TLCO are secured to make a parallel structure at each iteration. Based on the information storage capacity of the best solution in previous iterations, the constraints between the two processes are always established. These constraints will create two trends. The first trend is shown in the case of functions F1, F4, F6, in which the best solution is found after the first few iterations. This trend is reflected by the process of finding positions that are close to the current best solution with short distances whose values are controlled by step length S_*. And the second trend is illustrated in the case of functions F8, F14, F23, in which the best solution is found with much effort. Therefore, the frequency of appearance of reproductive termites will be registered with a higher probability than that of the functions F1, F4, F6. The probability distribution of the best solution in these functions is circumscribed in a wider region. Thus, the exploitation space of termites is likely to expand and more positions are spread over every best solution found at each iteration. The combination of soldier and termite workers in these functions shows the ability of TLCO to escape from local optima.

The ability of finding the new colony of reproductive termites is shown in the first column at the second row in Fig. 13. These termites appear to increase the opportunity of exploring new food sources and their effectiveness will support the termite workers and termite soldiers reset the search space. Then, based on the current best solution at each iteration, these search space will move to places having high probability to establish new search spaces. This process will be performed continuously through the over course of iterations. It leads to the evaluation of TLCO compared to the others optimization algorithms.

TLCO finds the best solution based on the self-improvement of the initial population compared to each termite in colony as shown in the second column in Fig. 13. This strategy will be more effective than the self-improvement of each termite in colony. At the same time, through a comparison with the individual population, the timing to abound the poor search spaces will be determined to reset a new search space.

The convergence curve shown in the third column at the second row in Fig. 13 is used to evaluate the convergence performance of TLCO. The convergence curves of F1, F4, F6 functions are very smooth and dropped rapidly, demonstrating that the skill of exploitation is more biased than the skill of exploration. In contrast, in the case of the remaining functions, whose curves are very rough and drop slowly, the skill of exploration is more biased than the skill of exploitation. Finally, the convergence curves can all accurately approximate the global optimum in the final iterations.

4.3. The comparison between TLCO with other algorithms

To evaluate the effectiveness of TLCO, eight well-known algorithms including GSA (Rashedi, et al., 2009), GWO (Mirjalili, et al., 2014), CS (X.-S. Yang & Deb, 2009), WOA (Mirjalili & Lewis, 2016), SCA (Mirjalili, 2016b), MFO (Mirjalili, 2015b), HHO (Heidari, et al., 2019) and AOA (Abualigah, Diabat, Mirjalili, Abd Elaziz, & Gandomi, 2021) are selected. The necessary parameters of all considered algorithms are set as follows Table 6.

At the beginning of this section, the convergence rate of TLCO and the other algorithms will be compared. For this test, all algorithms mentioned above have been implemented with the same initial conditions including the dimension of the search space $D = 30$, the number of particles $N = 30$ and the total number of iterations $K_{max} = 1000$. The historical values of the best fitness in whole the optimization algorithms is plotted in the same figure as shown in Fig. 14.

For the functions (F1-F13), these figures show that TLCO reaches a fast convergence rate and that it is successful in finding and exploiting the global optimum in almost all functions. The modulation of the

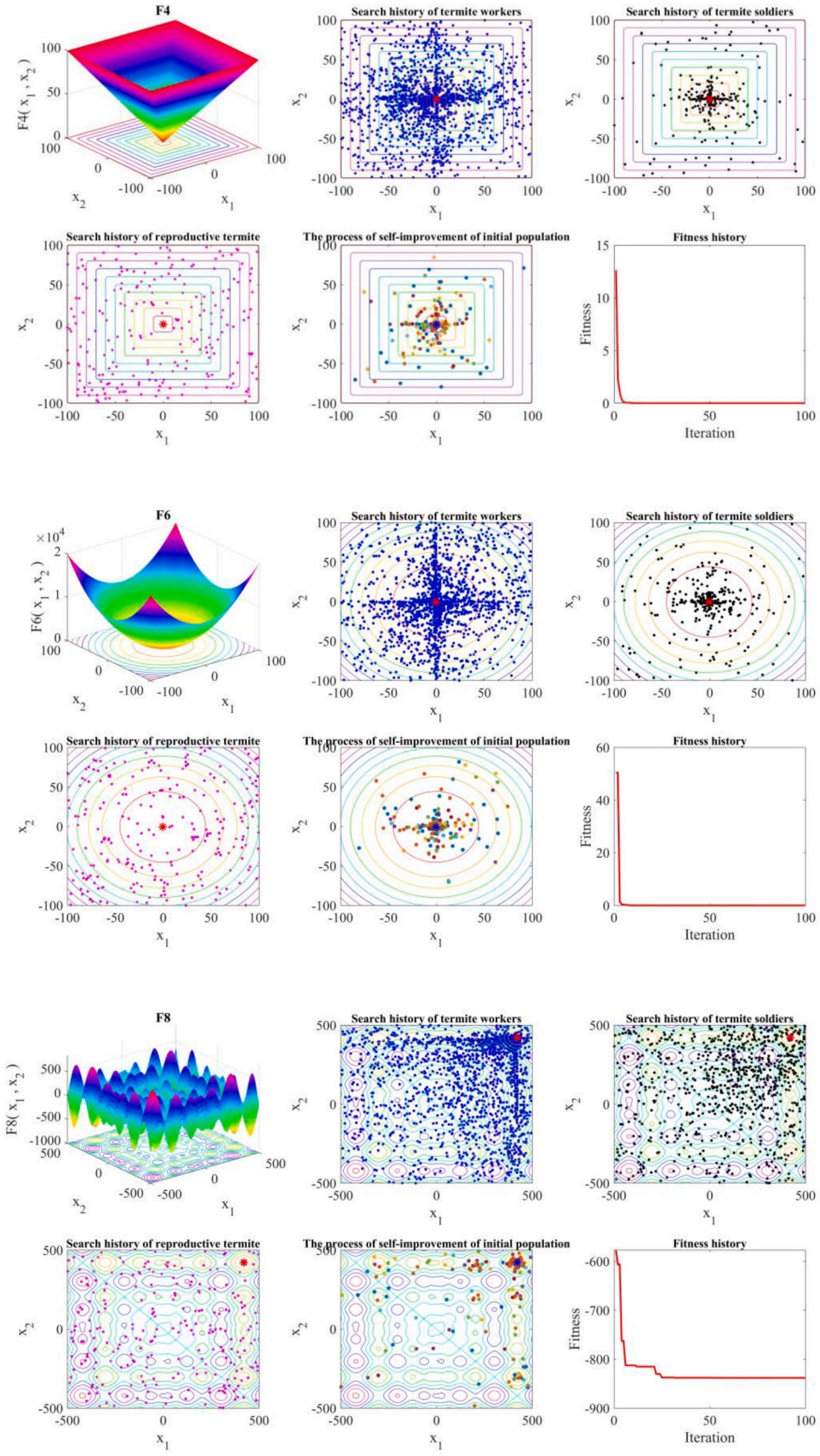

Fig. 13. (*continued*).

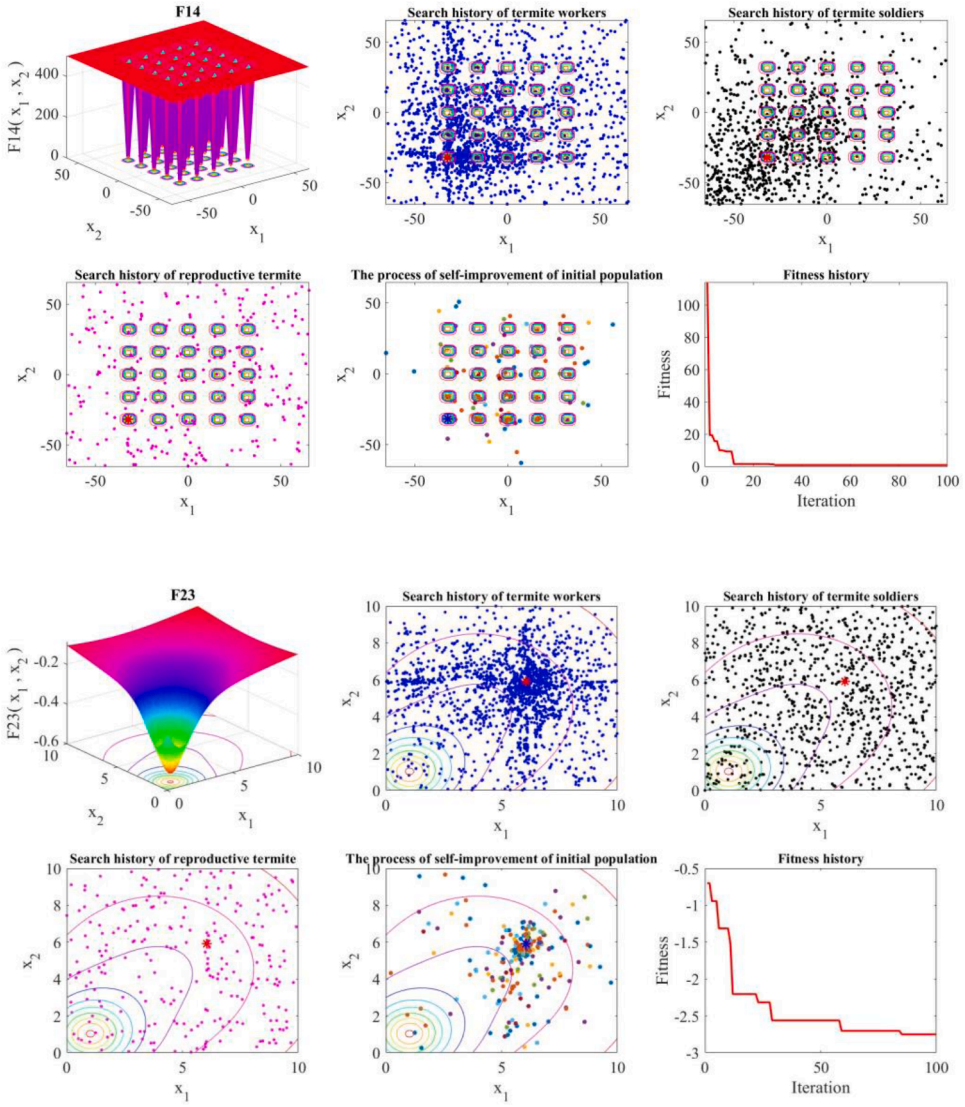

Fig. 13. (*continued*).

Table 6
Parameter settings for the other optimization algorithm.

Algorithm	Parameter	Value
TLCO	Free meta-heuristic optimization	None
GSA	Gravitational constant	$G = 100$
	Decreasing coefficient	$\beta = 50$
GWO	Convergence parameter	$\vec{a}$ linearly decreased from 2 to 0
	$\vec{r_1}$,$\vec{r_2}$	Random vector having value in range $[0,\ 1]$
CS	Fraction of worse nests	$p_a = 0.5$
	The parameter for calculating the step length S	$\beta = 1.5$
WOA	The shape of the logarithmic spiral	$b = 1$
	Convergence parameter	$\vec{a}$ linearly decreased from 2 to 0
	The random number parameter	$l = 1$
SCA	A constant for controlling the parameter $\vec{r_1}$	A = 2
MFO	Convergence parameter $\vec{a}$	linearly decreases from -1 to -2
	the shape of the logarithmic spiral	$b = 1$
HHO	The parameter for calculating the step length S	$\beta = 1.5$
	The initial energy	$E_0 = 2rand(0,1) - 1$
AOA	The sensitive parameter	$\alpha = 5$
	μ parameters	$\mu = 0.499$
	Minimum value of accelerated function	$Min = 0.2$
	Maximum value of accelerated function	$Max = 1$

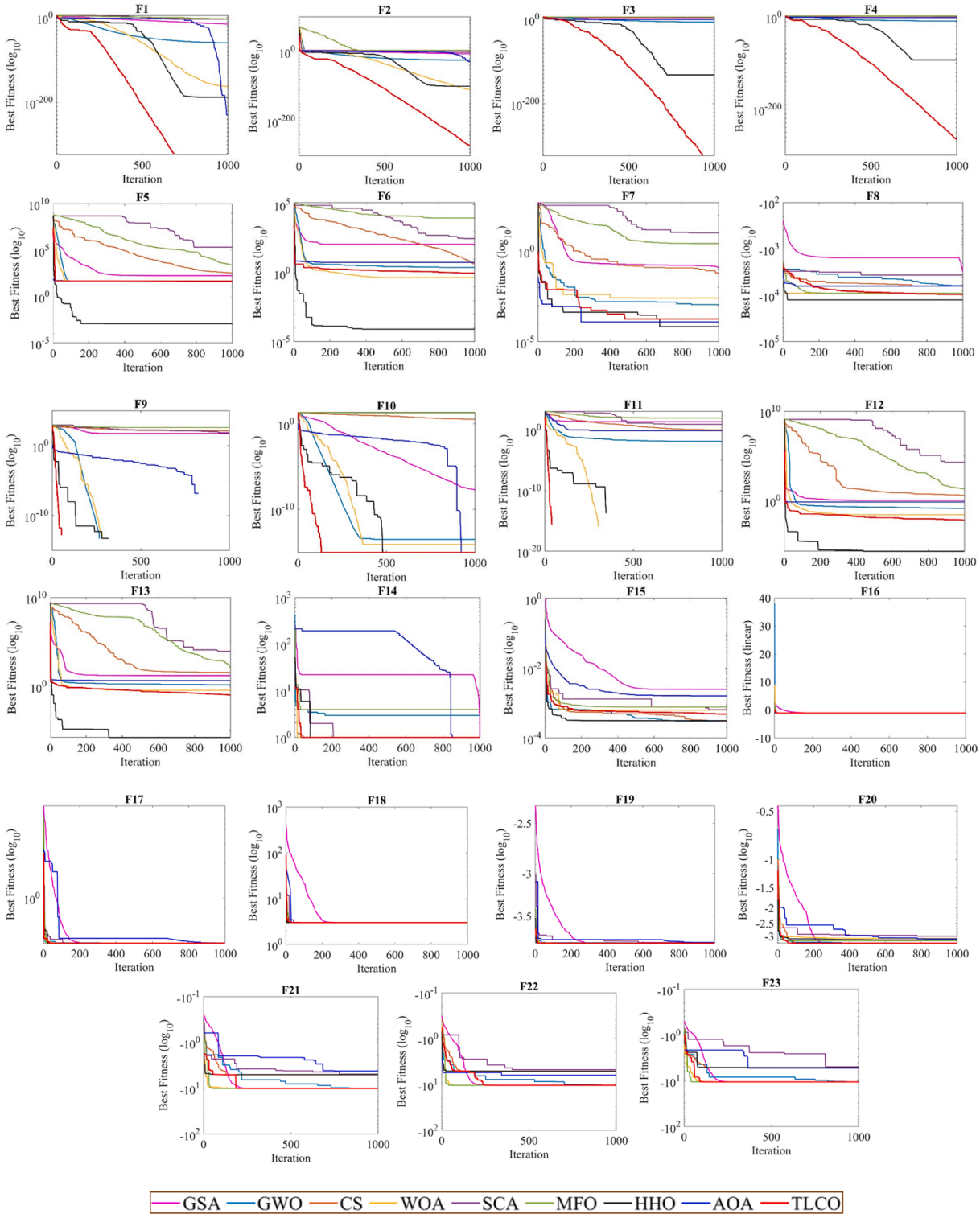

Fig. 14. Convergence trends of the 23 benchmark functions with different algorithms.

movement strategy implemented in TLCO provides powerful advantages to exploit the best solution. The processes of exploitation and exploration in TLCO are secured in parallel. As a result, TLCO shows better performances in convergence rate compared to other algorithms in the case of functions F1, F2, F3, F4, F5, F9, F10, F11 because of the long movements controlled in Eq. (6) in the first few iterations. However, TLCO fails to achieve the best convergence performance the case of functions F5, F6, F7, F8, F12, F13.

The common feature of functions F14-F23, which are multi-modal functions with low-dimension and a few local optima, is that almost all algorithms can exploit the best solution. However, the convergence rate of TLCO reaches stability faster for almost all functions compared to other algorithms with the exception of functions 21 and 23. Especially regarding the functions F14, F16, F18, F19 and F20, TLCO only requires a few iterations to achieve stability. This proves that the TLCO's ability to escape local optimal optimization is better than other algorithms that

Table 7
Comparison results of the 23 benchmark functions with different Algorithms.

Function	Different Algorithms with D = 30, D = 100, D = 1000 for F1-F13 and Fixed dimension for F14-F23								
		TLCO	GSA	GWO	CS	WOA	SCA	MFO	HHO	AOA
F1_Dim30	Mean	0.000E+00	5.524E-17	3.268E-59	6.254E-04	6.622E-146	3.813E-02	2.667E+03	1.338E-183	4.463E-30
	Rank	1	6	4	7	3	8	9	2	5
	Std	0.00E+00	1.11E-17	6.35E-59	3.37E-04	3.62E-145	1.64E-01	5.21E+03	0.00E+00	2.00E-29
F1_Dim100	Mean	0.000E+00	6.200E+02	8.791E-30	5.886E+02	3.506E-148	6.495E+03	3.212E+04	3.641E-178	2.109E-02
	Rank	1	7	4	6	3	8	9	2	5
	Std	0.00E+00	2.23E+02	9.70E-30	1.94E+02	1.09E-147	5.33E+03	1.06E+04	0.00E+00	8.57E-03
F1_Dim1000	Mean	0.00E+00	5.19E+02	8.72E-09	1.48E+05	3.25E-142	4.05E+05	2.49E+06	1.36E-180	1.65E+00
	Rank	1	6	4	7	3	8	9	2	5
	Std	0.00E+00	1.77E+02	3.33E-09	1.20E+04	1.03E-141	1.12E+05	4.52E+04	0.00E+00	7.17E-02
F2_Dim30	Mean	1.677E-262	1.326E+02	1.571E-33	1.214E+02	1.458E-103	8.161E-05	4.467E+02	2.484E-95	2.908E-11
	Rank	1	8	4	7	2	6	9	3	5
	Std	0.00E+00	3.99E+01	1.82E-33	2.19E+01	4.21E-103	1.98E-04	2.49E+02	1.10E-94	1.30E-10
F2_Dim100	Mean	1.178E-265	5.318E+00	6.010E-17	8.645E+02	2.161E-102	9.556E+00	2.450E+03	4.413E-93	1.267E-01
	Rank	1	6	4	8	2	7	9	3	5
	Std	0.00E+00	2.24E+00	2.92E-17	6.76E+01	7.18E-102	7.27E+00	4.60E+02	1.97E-92	1.07E-01
F2_Dim1000	Mean	1.41E-258	6.63E+00	2.32E+204	Inf	2.30E-100	Inf	Inf	1.23E-94	2.10E+01
	Rank	1	4	9	6	2	6	6	3	5
	Std	0.00E+00	1.92E+00	Inf	Inf	5.85E-100	Inf	Inf	3.45E-94	7.46E-01
F3_Dim30	Mean	0.000E+00	5.542E+02	3.362E-14	5.846E+02	1.787E+04	3.595E+03	1.882E+04	2.097E-141	2.244E-03
	Rank	1	5	3	6	8	7	9	2	4
	Std	0.00E+00	1.46E+02	1.69E-13	1.78E+02	9.91E+03	2.70E+03	1.29E+04	9.38E-141	5.33E-03
F3_Dim100	Mean	1.423E-248	8.567E+03	6.495E+00	3.620E+04	9.145E+05	1.902E+05	1.968E+05	1.853E-124	8.694E-01
	Rank	1	5	4	6	9	7	8	2	3
	Std	0.00E+00	2.29E+03	1.33E+01	8.63E+03	2.28E+05	5.25E+04	5.26E+04	8.27E-124	9.28E-01
F3_Dim1000	Mean	3.75E-07	9.55E+03	8.50E+05	5.03E+06	1.42E+08	2.44E+07	1.40E+07	2.82E-72	1.41E+02
	Rank	2	4	5	6	9	8	7	1	3
	Std	1.68E-06	2.07E+03	2.68E+05	1.16E+06	1.10E+08	4.86E+06	2.09E+06	1.23E-71	6.78E+01
F4_Dim30	Mean	3.914E-259	2.843E+00	1.306E-14	5.837E+00	3.929E+01	2.154E+01	6.740E+01	1.603E-95	2.610E-02
	Rank	1	5	3	6	8	7	9	2	4
	Std	0.00E+00	1.86E+00	1.44E-14	2.29E+00	3.02E+01	1.10E+01	8.93E+00	5.99E-95	2.03E-02
F4_Dim100	Mean	1.432E-247	1.529E+01	7.709E-03	2.157E+01	7.217E+01	8.720E+01	9.279E+01	4.778E-93	8.604E-02
	Rank	1	5	3	6	7	8	9	2	4
	Std	0.00E+00	1.45E+00	1.50E-02	1.61E+00	2.19E+01	3.48E+00	2.34E+00	1.82E-92	9.61E-03
F4_Dim1000	Mean	1.73E-228	1.54E+01	7.68E+01	3.91E+01	7.45E+01	9.96E+01	9.95E+01	7.45E-93	2.11E-01
	Rank	1	4	7	5	6	9	8	2	3
	Std	0.00E+00	1.26E+00	5.39E+00	2.03E+00	2.58E+01	9.34E-02	1.40E-01	2.04E-92	1.13E-02
F5_Dim30	Mean	2.721E+01	4.253E+01	2.687E+01	3.392E+01	2.719E+01	3.239E+02	2.156E+04	3.705E-03	2.818E+01
	Rank	4	7	2	6	3	8	9	1	5
	Std	9.85E-01	4.24E+01	9.34E-01	1.50E+01	4.52E-01	8.93E+02	3.84E+04	3.92E-03	3.44E-01
F5_Dim100	Mean	8.350E+00	1.612E+04	9.742E+01	3.466E+04	9.782E+01	4.614E+07	7.144E+07	7.860E-03	9.886E+01
	Rank	2	6	3	7	4	8	9	1	5
	Std	8.49E+00	1.12E+04	9.19E-01	1.61E+04	3.54E-01	2.11E+07	5.56E+07	1.40E-02	9.72E-02
F5_Dim1000	Mean	1.47E+02	1.98E+04	9.97E+02	4.09E+07	9.93E+02	2.94E+09	1.09E+10	1.24E-01	9.99E+02
	Rank	2	6	4	7	3	8	9	1	5
	Std	1.89E+02	1.28E+04	2.25E-01	6.83E+06	5.99E-01	9.85E+08	3.70E+08	1.79E-01	1.34E-01
F6_Dim30	Mean	3.855E-02	5.962E-17	6.396E-01	6.623E-04	8.689E-02	4.521E+00	2.337E+03	5.024E-05	2.792E+00
	Rank	4	1	6	3	5	8	9	2	7
	Std	2.46E-02	1.66E-17	3.81E-01	3.20E-04	1.09E-01	3.96E-01	4.31E+03	6.58E-05	3.41E-01
F6_Dim100	Mean	5.482E-01	1.483E+03	9.195E+00	6.414E+02	1.784E+00	5.568E+03	3.855E+04	2.126E-04	1.727E+01
	Rank	2	7	4	6	3	8	9	1	5
	Std	4.25E-01	4.43E+02	8.85E-01	1.31E+02	6.52E-01	3.64E+03	1.46E+04	2.68E-04	4.54E-01
F6_Dim1000	Mean	8.27E+00	1.85E+03	2.09E+02	1.45E+05	4.46E+01	3.65E+05	2.47E+06	1.64E-03	2.42E+02
	Rank	2	6	4	7	3	8	9	1	5
	Std	8.26E+00	6.47E+02	2.04E+00	1.31E+04	6.50E+00	1.61E+05	5.36E+04	2.62E-03	1.96E+00
F7_Dim30	Mean	2.445E-05	3.214E-02	9.002E-04	3.082E-02	1.734E-03	4.405E-02	3.333E+00	6.569E-05	2.759E-05
	Rank	1	7	4	6	5	8	9	3	2
	Std	1.87E-04	1.06E-02	3.83E-04	9.86E-03	1.90E-03	4.59E-02	6.72E+00	6.23E-05	2.76E-05
F7_Dim100	Mean	3.315E-04	2.632E+00	2.350E-03	3.052E-01	2.286E-03	7.128E+01	1.654E+02	6.390E-05	4.286E-05
	Rank	3	7	5	6	4	8	9	2	1
	Std	2.56E-04	1.04E+00	1.05E-03	4.97E-02	2.76E-03	5.83E+01	1.35E+02	5.47E-05	4.06E-05
F7_Dim1000	Mean	3.85E-04	2.15E+00	2.21E-02	3.45E+02	1.20E-03	4.28E+04	1.78E+05	6.57E-05	6.30E-05
	Rank	3	6	5	7	4	8	9	2	1
	Std	3.96E-04	6.17E-01	5.71E-03	3.48E+01	8.11E-04	9.44E+03	7.35E+03	5.50E-05	4.09E-05
F8_Dim30	Mean	−1.055E+04	−2.676E+03	−5.886E+03	−6.394E+03	−1.099E+04	−4.023E+03	−8.717E+03	−1.257E+04	−5.865E+03
	Rank	3	9	6	5	2	8	4	1	7
	Std	4.63E+02	4.89E+02	6.82E+02	3.18E+02	1.64E+03	3.13E+02	8.07E+02	1.69E-01	4.74E+02
F8_Dim100	Mean	−2.276E+04	−1.043E+03	−1.544E+04	−1.221E+04	−3.671E+04	−7.107E+03	−2.447E+04	−4.190E+04	−1.075E+04
	Rank	4	9	5	6	2	8	3	1	7
	Std	1.20E+03	3.16E+02	3.26E+03	7.25E+02	5.85E+03	3.38E+02	2.54E+03	6.08E-01	8.45E+02
F8_Dim1000	Mean	−9.92E+04	−8.88E+02	−9.78E+04	−3.89E+04	−3.77E+05	−2.27E+04	−1.01E+05	−4.19E+05	−3.51E+04
	Rank	4	9	5	6	2	8	3	1	7
	Std	6.82E+03	2.84E+02	7.18E+03	1.45E+03	5.63E+04	9.49E+02	6.69E+03	8.11E+00	2.52E+03
F9_Dim30	Mean	0.000E+00	1.788E+01	4.149E-01	5.486E+01	5.684E-15	2.105E+01	1.560E+02	0.000E+00	0.000E+00
	Rank	1	6	5	8	4	7	9	1	1
	Std	0.00E+00	4.64E+00	2.27E+00	1.42E+01	2.29E-14	2.83E+01	3.80E+01	0.00E+00	0.00E+00

(continued on next page)

Table 7 (*continued*)

Function	Different Algorithms with D = 30, D = 100, D = 1000 for F1-F13 and Fixed dimension for F14-*F*23								
		TLCO	GSA	GWO	CS	WOA	SCA	MFO	HHO	AOA
F9_Dim100	Mean	0.000E+00	1.357E+02	2.387E-13	2.613E+02	0.000E+00	2.264E+02	7.635E+02	0.000E+00	0.000E+00
	Rank	1	6	5	8	1	7	9	1	1
	Std	0.00E+00	1.40E+01	1.73E-13	3.12E+01	0.00E+00	9.55E+01	1.01E+02	0.00E+00	0.00E+00
F9_Dim1000	Mean	0.00E+00	1.38E+02	1.35E+01	7.41E+03	3.64E-13	1.72E+03	1.47E+04	0.00E+00	5.12E-05
	Rank	1	6	5	8	3	7	9	1	4
	Std	0.00E+00	1.59E+01	1.40E+01	1.71E+02	1.15E-12	9.30E+02	1.17E+02	0.00E+00	7.03E-06
F10_Dim30	Mean	8.882E-16	5.876E-09	1.534E-14	1.739E+00	4.323E-15	1.297E+01	1.582E+01	8.882E-16	8.882E-16
	Rank	1	6	5	7	4	8	9	1	1
	Std	0.00E+00	7.73E-10	3.09E-15	6.94E-01	2.18E-15	9.11E+00	6.20E+00	0.00E+00	0.00E+00
F10_Dim100	Mean	8.882E-16	3.002E+00	1.119E-13	8.055E+00	4.619E-15	1.667E+01	1.991E+01	8.882E-16	1.284E-04
	Rank	1	6	4	7	3	8	9	1	5
	Std	0.00E+00	4.38E-01	9.50E-15	9.01E-01	2.93E-15	6.70E+00	4.94E-02	0.00E+00	5.74E-04
F10_Dim1000	Mean	8.88E-16	3.01E+00	2.84E-06	1.31E+01	3.38E-15	1.60E+01	2.02E+01	8.88E-16	9.16E-03
	Rank	1	6	4	7	3	8	9	1	5
	Std	0.00E+00	5.81E-01	5.16E-07	3.34E-01	2.40E-15	5.23E+00	2.42E-01	0.00E+00	2.16E-04
F11_Dim30	Mean	0.000E+00	1.653E+01	1.412E-03	7.606E-02	1.907E-03	2.581E-01	2.100E+01	0.000E+00	6.015E-02
	Rank	1	8	3	6	4	7	9	1	5
	Std	0.00E+00	4.04E+00	4.63E-03	5.70E-02	1.04E-02	2.98E-01	3.00E+01	0.00E+00	4.89E-02
F11_dim100	Mean	0.000E+00	9.527E+01	5.328E-04	6.919E+00	5.710E-03	3.525E+01	2.242E+02	0.000E+00	9.709E-01
	Rank	1	8	3	6	4	7	9	1	5
	Std	0.00E+00	1.02E+01	2.38E-03	1.74E+00	2.55E-02	2.74E+01	9.46E+01	0.00E+00	1.08E-01
F11_Dim1000	Mean	0.00E+00	9.50E+01	2.87E-03	9.35E+02	0.00E+00	2.83E+03	1.56E+04	0.00E+00	6.70E+03
	Rank	1	5	4	6	1	7	9	1	8
	Std	0.00E+00	1.38E+01	6.05E-03	7.47E+01	0.00E+00	9.28E+02	2.98E+02	0.00E+00	7.33E+02
F12_Dim30	Mean	1.597E-03	1.374E-01	3.438E-02	1.416E+00	6.014E-03	7.778E+01	8.533E+06	1.002E-06	3.942E-01
	Rank	2	5	4	7	3	8	9	1	6
	Std	1.05E-03	2.88E-01	1.46E-02	8.58E-01	5.90E-03	2.97E+02	4.67E+07	1.45E-06	5.12E-02
F12_Dim100	Mean	2.351E-03	4.782E+00	2.468E-01	1.365E+01	2.303E-02	1.213E+08	1.585E+08	8.765E-07	8.566E-01
	Rank	2	6	4	7	3	8	9	1	5
	Std	2.76E-03	1.12E+00	6.67E-02	2.99E+00	3.04E-02	6.58E+07	1.44E+08	1.91E-06	1.60E-02
F12_Dim1000	Mean	4.03E-03	5.35E+00	8.58E-01	1.20E+06	5.70E-02	1.00E+10	2.68E+10	5.36E-07	1.10E+00
	Rank	2	6	4	7	3	8	9	1	5
	Std	3.79E-03	1.57E+00	2.30E-02	5.01E+05	2.63E-02	1.84E+09	1.12E+09	1.00E-06	8.14E-03
F13_Dim30	Mean	4.317E-03	6.080E-18	5.636E-01	5.612E-01	2.206E-01	6.885E+02	4.101E+07	2.237E-05	2.763E+00
	Rank	3	1	6	5	4	8	9	2	7
	Std	4.24E-03	1.82E-18	1.97E-01	1.39E+00	1.32E-01	3.74E+03	1.25E+08	3.87E-05	1.16E-01
F13_Dim100	Mean	9.121E-02	1.292E+02	6.405E+00	7.644E+02	1.874E+00	2.612E+08	2.595E+08	3.643E-05	9.922E+00
	Rank	2	6	4	7	3	9	8	1	5
	Std	9.17E-02	1.84E+01	4.57E-01	9.50E+02	1.00E+00	2.38E+08	2.63E+08	5.59E-05	6.68E-02
F13_Dim1000	Mean	9.08E-01	1.28E+02	6.37E+00	9.98E+02	2.07E+00	3.03E+08	2.38E+08	4.13E-04	1.00E+02
	Rank	2	6	4	7	3	9	8	1	5
	Std	9.56E-01	2.08E+01	4.77E-01	8.30E+02	6.40E-01	1.54E+08	2.15E+08	4.84E-04	3.99E-02
F14	Mean	0.998000	13.253000	3.641600	0.998000	3.058200	1.692500	2.400900	1.047719	9.310244
	Rank	1	9	7	1	6	4	5	3	8
	Std	2.33E-16	6.46E+00	4.01E+00	5.09E-17	3.42E+00	9.71E-01	4.34E+00	2.22E-01	4.31E+00
F15	Mean	3.75E-04	6.75E-03	5.55E-03	3.18E-04	6.80E-04	9.33E-04	9.87E-04	3.68E-04	3.18E-03
	Rank	3	9	8	1	4	5	6	2	7
	Std	8.90E-05	1.01E-02	8.78E-03	2.10E-05	3.98E-04	3.71E-04	3.78E-04	2.06E-04	7.27E-03
F16	Mean	−1.031628	−1.031628	−1.031628	−1.031628	−1.031628	−1.031628	−1.031628	−1.031628	−1.031628
	Rank	1	1	1	1	1	1	1	8	9
	Std	1.53E-16	8.82E-17	5.08E-09	1.84E-16	9.66E-11	2.00E-05	2.28E-16	2.81E-10	7.36E-08
F17	Mean	0.397887	0.397887	0.397887	0.397887	0.397887	0.397887	0.397887	0.397888	0.423211
	Rank	1	1	1	1	1	1	1	8	9
	Std	3.98E-16	0.00E+00	1.04E-06	5.16E-09	4.32E-07	5.75E-04	0.00E+00	4.02E-07	2.14E-02
F18	Mean	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000
	Rank	1	1	1	1	1	1	1	1	1
	Std	2.59E-15	1.08E-14	1.07E-05	5.49E-16	2.74E-05	1.69E-05	1.50E-15	5.77E-09	2.33E+01
F19	Mean	−3.862800	−3.862800	−3.861200	−3.862800	−3.857600	−3.855800	−3.862800	−3.861160	−3.852830
	Rank	1	1	5	1	7	8	1	6	9
	Std	2.02E-15	1.74E-15	2.94E-03	3.09E-07	5.92E-03	2.93E-03	2.28E-15	2.87E-03	2.96E-03
F20	Mean	−3.286327	−3.322000	−3.267900	−3.321900	−3.241800	−2.908200	−3.222500	−3.159485	−3.118269
	Rank	3	1	4	2	5	9	6	7	8
	Std	5.59E-02	1.76E-16	6.98E-02	3.56E-05	1.31E-01	3.28E-01	6.33E-02	8.93E-02	3.32E-02
F21	Mean	−9.795266	−7.912100	−9.647500	−8.132100	−8.122100	−2.663200	−6.265700	−5.284173	−4.002435
	Rank	1	5	2	3	4	9	6	7	8
	Std	1.14E+00	3.51E+00	1.56E+00	2.54E+00	2.91E+00	2.02E+00	3.39E+00	1.03E+00	1.27E+00
F22	Mean	−9.805793	−10.403000	−10.137000	−7.238400	−8.466700	−3.537900	−7.322600	−5.345458	−3.922557
	Rank	3	1	2	6	4	9	5	7	8
	Std	1.63E+00	1.35E-15	1.19E+00	2.65E+00	2.72E+00	2.16E+00	3.53E+00	1.16E+00	1.52E+00
F23	Mean	−10.260804	−10.201000	−9.859900	−8.123900	−8.732300	−4.832100	−8.601500	−5.127684	−4.175860
	Rank	1	2	3	6	4	8	5	7	9
	Std	1.21E+00	1.50E+00	2.13E+00	2.74E+00	2.87E+00	1.72E+00	3.45E+00	1.15E-03	1.66E+00
Sum		85	263	205	274	185	356	362	115	252
Mean rank		1.73	5.37	4.18	5.59	3.78	7.27	7.39	2.35	5.14
Final rank		1	6	4	7	3	8	9	2	5

need more iterations to achieve the necessary stability.

The second test is to gain the best value of two items including the mean values and the standard deviation values after 30 independent runs. An efficient algorithm must demonstrate its ability to find the best solution in search spaces of different dimensions D, especially, with large-scale problems. Therefore, in this work, the first 13 benchmark functions (F1-F13) including uni-modal benchmark functions and multi-modal benchmark functions have been selected with search space of dimensions $D = 30$, $D = 100$ and $D = 1000$. To provide an overview of the optimization algorithms, Friedman mean ranking test is employed to assess the performance of optimization algorithms as shown in each table.

Table 7 summarizes the results obtained for the different tested algorithms with functions (F1-F23). For $D = 30$, one can notice that TLCO performs better than other algorithms in the case of functions F1, F2, F3, F4, F9, F10. For the remaining functions, TLCO still reaches acceptable results and there is no large difference in comparison with the other algorithms. $D = 100$ shows a similar trend to the one observed in a search space of dimension $D = 30$. For even a higher dimension of the search space, when $D = 1000$, a common characteristic can be observed when the total number of iterations is set to 1000. In this case, the number of iterations is not large enough for algorithms to find the best global value with the exception of TLCO and HHO, whose performance is once again better. The results obtained using TLCO still achieve an acceptable accuracy. Meanwhile, most of the other algorithms fail to reach the best global value. TLCO marks a big improvement in finding the best solution in high-dimensional search space. It appears clearly that TLCO is efficient in high-dimension of the case studies. Usually, most algorithms will have difficulty in high-dimension search space because position updating does not guarantee a flexible movement.

Friedman ranking test has been carried for conducting ranking comparisons for the above-mentioned algorithms in a statistical way. The obtained results show that TLCO is ranked first compared to other comparative Algorithms following HHO and WOA.

5. CEC 2005 benchmark functions

In this section, we check the effectiveness of the TLCO algorithm in solving high-complex problems. Twenty benchmark functions representing different properties in CEC 2005 (Suganthan, et al., 2005) are used to evaluate the performance of TLCO. The properties of these functions range from simple to complex. They are summarized in Table 8, and Fig. 15 shows a 3D representation of some typical functions in CEC2005.

These benchmark functions are really high levels in comparison with the classical benchmark functions mentioned in the previous section because of their various complex properties and huge numbers of local optimal. A typical characteristic of almost optimization algorithms is that they are easy to get stuck at the local optima because the movement strategy is not flexible enough to approach the search space having a global optimum. Fig. 16 shows the ability of TLCO to escape from local optima for the functions CF16 with dimension investigated $D = 2$. TLCO just need 50 iterations to find the value of global optimum ($f_{global} = 120$). This can be achieved by the presence of reproductive termites, which will replace termite workers or termite soldier trapped too long in a poor search space. It leads to a wide distribution search space of TLCO and this is the superiority of the TLCO compared to other algorithms.

To evaluate the performance of algorithms for solving these benchmark functions in a large-scale dimension, TLCO and other algorithms are set with the same dimension $D = 30$, $D = 50$. The number of particles is $N = 50$ and the total number of iterations is $K_{\max} = 5000$.

Two metrics including the mean value and the standard deviation are computed for each algorithm with 10 independent runs as shown in Table 9. Friedman mean ranking test is employed to assess the general performance of optimization algorithms as shown in each table. At the same time to assess the significant difference between TLCO and the individual algorithm, Wilcoxon rank-sum test, for which the significance level of α is selected 0.05, is used. Thus, the performance of TLCO and peer algorithms is compared on the results of the values of p_{value}. If $p_{value} > \alpha$, there is not significant difference between the pair of algorithms. If $p_{value} < \alpha$, the significant difference is registered between the pair of algorithms. The statistics tests using Wilcoxon rank-sum test is shown in Table 10. Once again, TLCO still proves its reliability and effectiveness in solving high complexity functions. TLCO still achieves the best performance based on Friedman mean ranking test followed by CS, GSA, GWO, HHO, MFO, WOA, SCA and AOA. For functions CF1, CF2, CF9, CF13, CF18, CF19 and CF20, TLCO finds the best solution for both $D = 30$ and $D = 50$ compared to the other algorithms. For the remaining function, although TLCO failed to find the best solution, the values exploited by TLCO still have acceptable errors and there is no

Table 8
Descriptions of benchmark functions in CEC2005.

Functions	Description	Dimension	Solution space	f_{min}
Unimodal functions				
CF1	Shifted Sphere Function	D	$[-100, 100]$	−450
CF2	Shifted Schwefel's Problem 1.2	D	$[-100, 100]$	−450
CF3	Shifted Rotated High Conditioned Elliptic Function	D	$[-100, 100]$	−450
CF4	Shifted Schwefel's Problem 1.2 with Noise in Fitness	D	$[-100, 100]$	−450
CF5	Schwefel's Problem 2.6 with Global Optimum on Bounds	D	$[-100, 100]$	−310
Multimodal functions				
CF6	Shifted Rosenbrock's Function	D	$[-100, 100]$	−390
CF7	Shifted Rotated Griewank's Function without Bounds	D	$[0, 600]$	−180
CF8	Shifted Rotated Ackley's Function with Global Optimum on Bounds	D	$[-32, 32]$	−140
CF9	Shifted Rastrigin's Function	D	$[-5, 5]$	−330
CF10	Shifted Rotated Rastrigin's Function	D	$[-5, 5]$	−330
CF11	Shifted Rotated Weierstrass Function	D	$[-0.5, 0.5]$	90
CF12	Schwefel's Problem 2.13	D	$[-\pi, \pi]$	−460
Expanded functions				
CF13	Shifted Expanded Griewank's plus Rosenbrock's Function	D	$[-3, 1]$	−130
CF14	shifted Rotated Expanded Scaffer's CF6 Function	D	$[-100, 100]$	−300
Hybrid composite functions				
CF15	Hybrid Composition Function	D	$[-5, 5]$	120
CF16	Rotated Version of Hybrid Composition Function CF15	D	$[-5, 5]$	120
CF17	Rotated Hybrid Composition Function with Noise in Fitness	D	$[-5, 5]$	120
CF18	Rotated Hybrid Composition Function	D	$[-5, 5]$	10
CF19	Rotated Hybrid Composition Function with a Narrow Basin for the Global Optimum	D	$[-5, 5]$	10
CF20	Rotated Hybrid Composition Function with Global Optimum on the Bounds	D	$[-5, 5]$	10

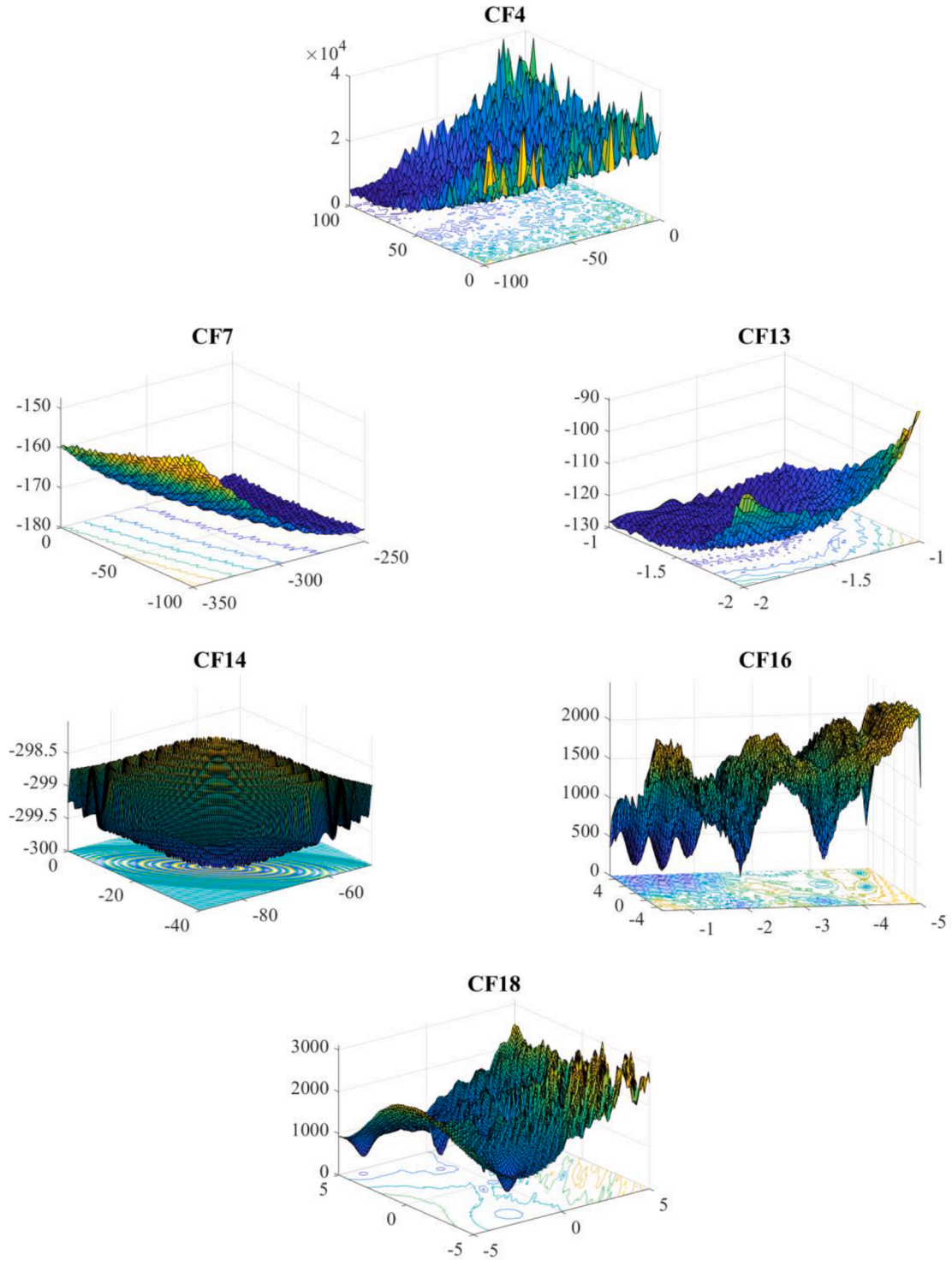

Fig. 15. 3D visualization of typical functions in CEC2005.

significant difference compared to other algorithms. For Wilcoxon rank-sum test, this method is applied for the single problem. Based on the corresponding statistical results for each function, TLCO shows superiority compared to the others algorithms. Meanwhile, CS and GSA are algorithms that registers the 2nd rank and 3rd rank. It can be seen that their abilities for exploration of the new search space are appreciated in comparison with WOA, SCA and MFO, HHO, AOA. GSA creates a new search space by mutating the current position with other position randomly selected in the population size. Meanwhile, CS uses the step length to expand the search space. In other words, the algorithms, which have strategies to create search space diversities, will register a high probability of exploiting the best solution in problems having a huge local optimal. Overall, TLCO still has a better performance than GSA, and CS algorithms based on the emergence of reproductive termites. They will create a new search space spread throughout boundary conditions of function problems. The computation time of TLCO is not advantageous as WOA, GWO algorithms. However, it can be seen that there is no large difference between TLCO and these algorithms. The computation time of TLCO can compete with that of the other algorithms as shown in Table 9.

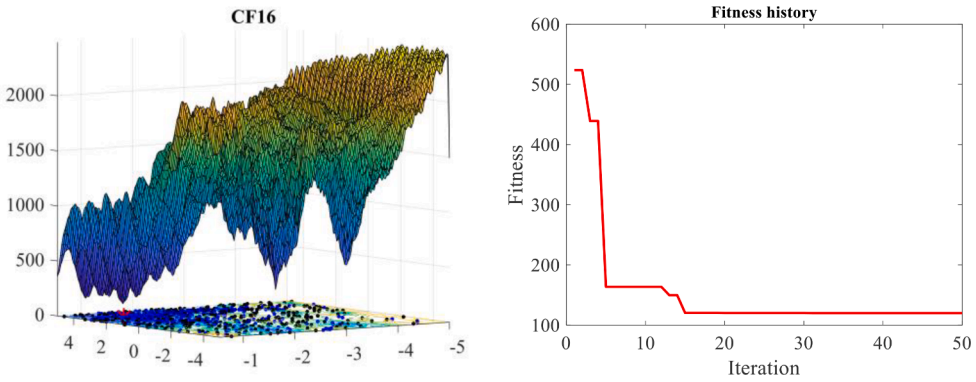

Fig. 16. The ability of escaping local optimal of TLCO with 50 iterations.

Table 9
Comparison results of 20 CEC2005 benchmark functions with dimension $D = 30$ and $D = 50$.

Function	Different Algorithms								
		TLCO	GSA	GWO	CS	WOA	SCA	MFO	HHO	AOA
CF1_Dim30	Mean	−4.500E+02	−4.500E+02	6.703E+02	−4.500E+02	−4.493E+02	1.108E+04	6.519E+03	−4.434E+02	8.684E+04
	Rank	1	1	6	1	4	8	7	5	9
	Std	7.654E-06	1.992E-14	5.643E+02	5.992E-14	2.039E-01	1.625E+03	6.515E+03	8.544E-01	5.079E+03
CF1_Dim50	Mean	−4.500E+02	−4.500E+02	3.369E+03	−4.500E+02	−4.356E+02	4.280E+04	2.597E+04	−4.210E+02	1.476E+05
	Rank	1	1	6	1	4	8	7	5	9
	Std	3.259E-06	7.339E-14	2.519E+03	5.729E-12	3.026E+00	4.926E+03	1.074E+04	4.887E+00	3.068E-11
CF2_Dim30	Mean	−3.858E+02	2.690E+03	8.288E+03	3.019E+02	5.907E+04	1.923E+04	1.395E+04	−6.268E+00	1.616E+05
	Rank	1	4	5	3	8	7	6	2	9
	Std	5.859E+01	3.565E+02	4.506E+03	1.259E+02	8.560E+03	3.496E+03	1.771E+04	7.816E+01	4.278E+04
CF2_Dim50	Mean	5.791E+03	1.079E+04	2.371E+04	1.968E+04	1.622E+05	6.305E+04	7.503E+04	6.495E+03	4.319E+05
	Rank	1	3	5	4	8	6	7	2	9
	Std	1.444E+03	1.087E+03	4.711E+03	1.626E+03	2.451E+04	7.015E+03	3.859E+04	8.667E+02	1.546E+05
CF3_Dim30	Mean	7.527E+06	1.831E+06	1.809E+07	1.012E+07	2.502E+07	1.488E+08	2.336E+07	9.098E+06	2.362E+09
	Rank	2	1	5	4	7	8	6	3	9
	Std	2.548E+06	2.857E+05	7.937E+06	2.979E+06	7.637E+06	5.028E+07	3.354E+07	4.370E+06	6.623E+08
CF3_Dim50	Mean	2.887E+07	1.302E+07	6.049E+07	4.994E+07	1.015E+08	8.370E+08	1.580E+08	2.488E+07	1.004E+10
	Rank	3	1	5	4	6	8	7	2	9
	Std	9.109E+06	1.183E+07	1.609E+07	1.167E+07	2.271E+07	1.730E+08	1.208E+08	4.838E+06	1.896E+09
CF4_Dim30	Mean	1.027E+04	2.649E+04	1.551E+04	1.602E+04	1.422E+05	2.586E+04	2.409E+04	2.825E+04	2.099E+05
	Rank	1	6	2	3	8	5	4	7	9
	Std	2.534E+03	1.569E+03	4.479E+03	3.071E+03	2.961E+04	4.648E+03	1.033E+04	5.789E+03	8.228E+04
CF4_Dim50	Mean	4.301E+04	8.320E+04	3.422E+04	7.120E+04	4.831E+05	7.349E+04	1.088E+05	6.202E+04	5.661E+05
	Rank	2	6	1	4	8	5	7	3	9
	Std	9.472E+03	1.448E+04	5.464E+03	9.704E+03	1.411E+05	1.256E+04	3.588E+04	8.353E+03	1.827E+05
CF5_Dim30	Mean	6.358E+03	6.876E+03	4.403E+03	5.025E+03	1.579E+04	1.535E+04	9.145E+03	1.160E+04	4.632E+04
	Rank	3	4	1	2	8	7	5	6	9
	Std	1.243E+03	1.947E+03	2.771E+03	9.103E+02	3.440E+03	1.819E+03	1.567E+03	1.969E+03	4.179E+03
CF5_Dim50	Mean	1.513E+04	1.393E+04	1.133E+04	1.685E+04	3.070E+04	2.711E+04	2.292E+04	2.169E+04	5.145E+04
	Rank	3	2	1	4	8	7	6	5	9
	Std	2.596E+03	2.008E+03	1.944E+03	9.796E+02	3.342E+03	1.845E+03	3.956E+03	2.820E+03	4.007E+03
CF6_Dim30	Mean	1.333E+03	6.768E+02	4.747E+07	4.075E+02	1.780E+04	9.173E+08	1.570E+09	2.881E+03	4.404E+10
	Rank	3	2	6	1	5	7	8	4	9
	Std	1.929E+03	2.050E+02	9.485E+07	4.250E+00	1.480E+04	2.393E+08	1.584E+09	1.493E+03	7.033E+08
CF6_Dim50	Mean	2.953E+03	7.785E+02	6.088E+08	4.309E+02	1.595E+05	8.241E+09	1.684E+10	1.286E+04	6.616E+10
	Rank	3	2	6	1	5	7	8	4	9
	Std	2.413E+03	2.120E+02	6.753E+08	6.565E+00	8.890E+04	1.897E+09	1.414E+10	3.767E+03	3.049E+08
CF7_Dim30	Mean	4.523E+03	8.232E+03	4.516E+03	1.123E+04	4.545E+03	4.555E+03	4.522E+03	4.517E+03	1.169E+04
	Rank	4	7	1	8	5	6	3	2	9
	Std	2.247E+00	1.847E+02	6.771E-03	6.106E+02	5.026E+01	4.748E+01	1.746E+01	7.533E-01	6.219E+02
CF7_Dim50	Mean	6.053E+03	1.372E+04	6.022E+03	1.647E+04	6.244E+03	6.198E+03	6.062E+03	6.021E+03	1.648E+04
	Rank	4	7	1	8	5	6	3	2	9
	Std	1.533E+01	6.117E+02	8.238E+00	6.814E+02	7.965E+01	5.337E+01	8.159E+01	2.699E+00	9.252E+02
CF8_Dim30	Mean	−1.197E+02	−1.199E+02	−1.190E+02	−1.194E+02	−1.192E+02	−1.190E+02	−1.195E+02	−1.194E+02	−1.188E+02
	Rank	2	1	7	5	6	7	3	4	9
	Std	8.343E-02	3.992E-02	2.425E-02	4.430E-02	6.622E-02	6.009E-02	1.756E-01	1.066E-01	1.053E-01
CF8_Dim50	Mean	−1.196E+02	−1.199E+02	−1.189E+02	−1.190E+02	−1.190E+02	−1.188E+02	−1.196E+02	−1.193E+02	−1.186E+02
	Rank	2	1	7	6	5	8	3	4	9
	Std	6.682E-02	2.424E-02	3.422E-02	4.289E-02	9.693E-02	4.695E-02	1.298E-01	9.967E-02	5.640E-02
CF9_Dim30	Mean	−3.262E+02	−3.146E+02	−2.380E+02	−3.250E+02	−1.047E+02	−8.085E+01	−2.073E+02	−1.090E+02	1.781E+02
	Rank	1	3	4	2	5				
	Std	2.336E+00	6.621E+00	2.506E+01	8.852E-01	4.161E+01	2.520E+01	3.304E+01	2.134E+01	1.229E+01
CF9_Dim50	Mean	−3.062E+02	−2.524E+02	−1.108E+02	−2.666E+02	8.124E+01	2.011E+02	−4.988E+01	1.140E+02	5.772E+02
	Rank	1	3	4	2	6	8	5	7	9
	Std	4.694E+00	9.143E+00	2.752E+01	6.240E+00	5.710E+01	4.316E+01	5.397E+01	5.890E+01	2.736E+00

(*continued on next page*)

Table 9 (*continued*)

Function	Different Algorithms								
		TLCO	GSA	GWO	CS	WOA	SCA	MFO	HHO	AOA
CF10_Dim30	Mean	−1.483E+02	−2.964E+02	−2.117E+02	−8.269E+01	8.237E+01	2.150E+01	−1.237E+02	5.749E+01	5.307E+02
	Rank	3	1	2	5	8	6	4	7	9
	Std	6.606E+01	3.869E+00	5.130E+01	3.079E+01	5.416E+01	1.582E+01	5.512E+01	9.263E+01	8.268E+01
CF10_Dim50	Mean	3.118E+01	−2.743E+02	−3.811E+01	2.160E+02	5.017E+02	4.457E+02	1.563E+02	5.243E+02	1.061E+03
	Rank	3	1	2	5	7	6	4	8	9
	Std	5.986E+01	8.028E+00	1.160E+02	5.254E+01	1.613E+02	4.966E+01	1.060E+02	1.155E+02	2.397E-13
CF11_Dim30	Mean	1.179E+02	9.000E+01	1.067E+02	1.146E+02	1.275E+02	1.303E+02	1.207E+02	1.268E+02	1.375E+02
	Rank	4	1	2	3	7	8	5	6	9
	Std	5.672E+00	7.019E-05	1.743E+00	1.754E+00	2.233E+00	9.545E-01	4.391E+00	2.805E+00	2.686E+00
CF11_Dim50	Mean	1.465E+02	9.031E+01	1.251E+02	1.406E+02	1.587E+02	1.644E+02	1.485E+02	1.596E+02	1.751E+02
	Rank	4	1	2	3	6	8	5	7	9
	Std	3.858E+00	5.574E-01	4.300E+00	2.122E+00	3.101E+00	1.090E+00	3.664E+00	4.156E+00	2.711E+00
CF12_Dim30	Mean	2.218E+04	1.370E+03	6.993E+05	3.100E+03	1.816E+05	9.102E+05	1.624E+05	3.105E+05	2.301E+06
	Rank	3	1	7	2	5	8	4	6	9
	Std	1.264E+04	2.748E+03	4.446E+05	1.145E+03	1.115E+05	1.060E+05	8.162E+04	9.555E+04	2.443E+05
CF12_Dim50	Mean	1.401E+05	8.238E+03	3.141E+06	5.125E+04	1.565E+06	5.030E+06	9.890E+05	2.048E+06	9.738E+06
	Rank	3	1	7	2	5	8	4	6	9
	Std	3.616E+04	5.327E+03	2.349E+06	1.328E+04	4.554E+05	5.845E+05	2.782E+05	3.418E+05	7.717E+05
CF13_Dim30	Mean	−1.282E+02	−1.260E+02	−1.259E+02	−1.248E+02	−1.108E+02	−9.844E+01	−8.351E+01	−9.746E+01	1.692E+03
	Rank	1	2	3	4	5	6	8	7	9
	Std	3.993E-01	1.063E+00	1.198E+00	4.817E-01	7.725E+00	6.096E+00	4.283E+01	4.803E+00	1.000E+03
CF13_Dim50	Mean	−1.248E+02	−1.221E+02	−1.150E+02	−1.119E+02	−7.846E+01	−2.319E+01	2.499E-01	−5.994E+01	5.471E+03
	Rank	1	2	3	4	5	7	8	6	9
	Std	1.501E+00	1.479E+00	4.711E+00	1.199E+00	1.284E+01	2.879E+01	1.431E+02	8.893E+00	1.309E+03
CF14_Dim30	Mean	−2.874E+02	−2.859E+02	−2.884E+02	−2.872E+02	−2.867E+02	−2.866E+02	−2.864E+02	−2.867E+02	−2.857E+02
	Rank	2	8	1	3	4	6	7	5	9
	Std	2.284E-01	2.685E-01	5.924E-01	4.091E-01	4.666E-01	1.444E-01	5.048E-01	1.885E-01	2.069E-01
CF14_Dim50	Mean	−2.773E+02	−2.760E+02	−2.785E+02	−2.774E+02	−2.770E+02	−2.768E+02	−2.767E+02	−2.770E+02	−2.758E+02
	Rank	3	8	1	2	5	6	7	4	9
	Std	2.075E-01	2.056E-01	5.004E-01	2.076E-01	3.576E-01	1.985E-01	2.721E-01	4.616E-01	1.891E-01
CF15_Dim30	Mean	4.841E+02	4.355E+02	6.239E+02	2.360E+02	7.625E+02	8.347E+02	6.419E+02	7.409E+02	1.430E+03
	Rank	3	2	4	1	7	8	5	6	9
	Std	6.603E+01	1.011E+01	9.686E+01	2.039E+01	2.101E+02	8.895E+01	9.155E+01	4.671E+01	1.154E+02
CF15_Dim50	Mean	4.921E+02	5.200E+02	5.586E+02	3.337E+02	9.877E+02	9.693E+02	6.982E+02	8.130E+02	1.579E+03
	Rank	2	3	4	1	8	7	5	6	9
	Std	1.479E+02	2.210E-13	5.531E+01	3.354E+01	1.842E+02	6.756E+01	5.058E+01	1.678E+02	7.025E+01
CF16_Dim30	Mean	3.617E+02	4.896E+02	4.120E+02	3.545E+02	6.041E+02	5.377E+02	3.860E+02	5.590E+02	1.378E+03
	Rank	2	5	4	1	8	6	3	7	9
	Std	1.612E+02	1.138E+02	1.798E+02	3.897E+01	6.520E+01	5.664E+01	5.153E+01	1.032E+02	8.195E+01
CF16_Dim50	Mean	4.518E+02	5.737E+02	4.312E+02	4.325E+02	7.245E+02	6.975E+02	4.603E+02	6.349E+02	1.498E+03
	Rank	3	5	1	2	8	7	4	6	9
	Std	8.054E+01	7.854E+00	1.502E+02	2.492E+01	1.293E+02	1.169E+02	6.585E+01	8.913E+01	9.878E+01
CF17_Dim30	Mean	4.516E+02	5.090E+02	4.574E+02	3.438E+02	6.634E+02	6.010E+02	4.155E+02	6.291E+02	1.548E+03
	Rank	3	5	4	1	8	6	2	7	9
	Std	1.635E+02	1.788E+02	2.093E+02	2.127E+01	7.591E+01	7.204E+01	6.845E+01	1.100E+02	1.713E+02
CF17_Dim50	Mean	4.806E+02	2.470E+02	3.996E+02	4.960E+02	7.650E+02	7.011E+02	4.936E+02	6.428E+02	1.693E+03
	Rank	3	1	2	5	8	7	4	6	9
	Std	6.025E+01	1.248E+02	9.641E+01	2.808E+01	9.681E+01	7.539E+01	6.615E+01	4.270E+01	1.017E+02
CF18_Dim30	Mean	9.100E+02	9.463E+02	9.438E+02	9.208E+02	1.035E+03	1.030E+03	9.249E+02	9.100E+02	9.100E+02
	Rank	1	7	6	4	9	8	5	1	1
	Std	0.000E+00	3.117E+01	1.716E+01	1.187E+00	1.198E+02	1.555E+01	7.133E+00	0.000E+00	0.000E+00
CF18_Dim50	Mean	9.100E+02	9.507E+02	9.881E+02	9.462E+02	1.105E+03	1.166E+03	1.015E+03	9.100E+02	9.100E+02
	Rank	1	5	6	4	8	9	7	1	1
	Std	0.000E+00	2.417E+01	7.514E+00	3.617E+00	1.780E+02	1.817E+01	3.445E+01	0.000E+00	0.000E+00
CF19_Dim30	Mean	9.100E+02	9.610E+02	9.61E+02	9.220E+02	1.032E+03	1.033E+03	9.367E+02	9.100E+02	9.100E+02
	Rank	1	6	7	4	8	9	5	1	1
	Std	0.000E+00	2.018E+01	2.02E+01	1.808E+00	6.578E+01	1.509E+01	2.574E+01	0.000E+00	0.000E+00
CF19_Dim50	Mean	9.100E+02	9.507E+02	9.857E+02	9.483E+02	1.157E+03	1.159E+03	1.020E+03	9.100E+02	9.100E+02
	Rank	1	5	6	4	8	9	7	1	1
	Std	0.000E+00	2.145E+01	1.543E+01	5.993E+00	1.772E+02	2.632E+01	5.269E+01	0.000E+00	0.000E+00
CF20_Dim30	Mean	9.100E+02	9.349E+02	9.35E+02	9.219E+02	1.032E+03	1.033E+03	9.386E+02	9.100E+02	9.100E+02
	Rank	1	5	6	4	8	9	7	1	1
	Std	0.000E+00	3.843E+01	3.84E+01	1.749E+00	6.578E+01	1.509E+01	1.797E+01	0.000E+00	0.000E+00
CF20_Dim50	Mean	9.100E+02	1.037E+03	9.516E+02	9.223E+02	1.042E+03	1.029E+03	9.323E+02	9.100E+02	9.100E+02
	Rank	1	8	6	4	9	7	5	1	1
	Std	0.000E+00	5.595E+01	1.088E+01	1.178E+00	6.078E+01	1.311E+01	2.319E+01	0.000E+00	0.000E+00
Sum		87	138	159	131	265	279	210	173	303
Mean rank		2.175	3.45	3.975	3.275	6.625	6.975	5.25	4.325	7.575
Final rank		1	3	4	2	7	8	6	5	9
Total time (s)_Dim30 for the first run (F1-F20)		5.16E+03	4.68E+04	4.79E+03	1.52E+04	4.52E+03	2.19E+04	2.53E+04	8.11E+03	1.15E+04
Total time (s)_Dim50 for the first run (F1-F20)		5.45E+03	4.97E+04	5.08E+03	1.67E+04	4.81E+03	2.41E+04	2.78E+04	8.92E+03	1.26E+04

Table 10
p_{value} of Wilcoxon rank sum test between TLCO and the other algorithms for solving CEC2005 functions at$\alpha = 0.05$.

Function	Different Algorithms CEC2005_Dim30 and Dim50																	
TLCO vs	GSA		GWO		CS		WOA		SCA		MFO		HHO		AOA			
CF1_Dim30	1.11E-04	~	1.83E-04	+	6.39E-05		1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.32E-04	+		
CF1_Dim50	1.49E-04	~	1.83E-04	+	1.82E-04		1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	6.39E-05	+		
CF2_Dim30	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+		
CF2_Dim50	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.21E-01	~	1.83E-04	+		
CF3_Dim30	5.83E-04	–	3.61E-03	+	1.71E-03	+	1.83E-04	+	1.83E-04	+	4.27E-01	~	4.73E-01	~	1.83E-04	+		
CF3_Dim50	1.13E-02	–	7.69E-04	+	1.01E-03	+	1.83E-04	+	1.83E-04	+	3.12E-02	+	3.07E-01	~	1.83E-04	+		
CF4_Dim30	1.83E-04	+	4.27E-01	~	7.69E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+		
CF4_Dim50	1.83E-04	+	4.52E-02	–	3.30E-04	+	1.83E-04	+	2.46E-04	+	2.46E-04	+	1.01E-03	+	1.83E-04	+		
CF5_Dim30	6.78E-01	~	2.11E-02	–	3.07E-01	~	1.83E-04	+	1.83E-04	+	1.40E-01	~	4.40E-04	+	1.83E-04	+		
CF5_Dim50	2.41E-01	~	2.83E-03	–	3.76E-02	+	1.83E-04	+	1.83E-04	+	5.83E-04	+	5.83E-04	+	1.83E-04	+		
CF6_Dim30	9.10E-01	~	1.83E-04	+	1.83E-04	–	4.40E-04	+	1.83E-04	+	1.83E-04	+	9.11E-03	+	1.32E-04	+		
CF6_Dim50	4.40E-04	–	1.83E-04	+	1.83E-04	–	1.83E-04	+	1.83E-04	+	1.83E-04	+	2.46E-04	+	1.32E-04	+		
CF7_Dim30	1.83E-04	+	1.83E-04	–	1.83E-04	+	2.97E-04	+	5.83E-04	+	2.12E-02	–	1.83E-04	–	1.83E-04	+		
CF7_Dim50	1.83E-04	+	1.83E-04	–	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.40E-01	~	1.83E-04	–	1.83E-04	+		
CF8_Dim30	1.83E-04	–	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	3.85E-01	~	3.30E-04	+	1.83E-04	+		
CF8_Dim50	1.83E-04	–	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	9.10E-01	~	1.83E-04	+	1.83E-04	+		
CF9_Dim30	1.82E-04	+	1.83E-04	+	4.59E-03	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.32E-04	+		
CF9_Dim50	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	8.75E-05	+		
CF10_Dim30	1.82E-04	–	1.21E-01	~	4.52E-02	+	4.40E-04	+	3.30E-04	+	3.76E-02	+	4.40E-04	+	1.81E-04	+		
CF10_Dim50	1.83E-04	–	6.40E-02	~	1.83E-04	+	1.83E-04	+	1.83E-04	+	2.83E-03	+	1.83E-04	+	6.39E-05	+		
CF11_Dim30	1.83E-04	–	7.69E-04	–	1.40E-02	–	1.71E-03	+	1.83E-04	+	1.40E-01	~	7.69E-04	+	1.83E-04	+		
CF11_Dim50	1.83E-04	–	1.83E-04	–	2.20E-03	–	1.83E-04	+	1.83E-04	+	3.85E-01	~	1.83E-04	+	1.83E-04	+		
CF12_Dim30	1.83E-04	–	1.83E-04	+	1.83E-04	–	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+		
CF12_Dim50	1.83E-04	–	1.83E-04	+	1.83E-04	–	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+		
CF13_Dim30	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+		
CF13_Dim50	1.71E-03	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+		
CF14_Dim30	1.83E-04	+	2.20E-03	–	2.41E-01	~	4.59E-03	+	1.83E-04	+	2.20E-03	+	3.30E-04	+	1.83E-04	+		
CF14_Dim50	1.83E-04	+	1.83E-04	–	2.12E-01	~	1.21E-01	~	3.30E-04	+	1.31E-03	+	1.04E-01	~	1.83E-04	+		
CF15_Dim30	1.73E-02	–	9.70E-01	~	2.83E-03	–	1.31E-03	+	1.83E-04	+	1.73E-02	+	1.83E-04	+	1.83E-04	+		
CF15_Dim50	1.36E-01	~	8.90E-02	~	1.01E-03	–	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+	1.83E-04	+		
CF16_Dim30	4.57E-03	+	2.73E-01	~	4.73E-01	~	6.40E-02	~	3.45E-01	~	5.71E-01	~	1.62E-01	~	1.83E-04	+		
CF16_Dim50	1.83E-04	+	4.27E-01	~	3.12E-02	–	1.83E-04	+	1.83E-04	+	9.11E-03	+	1.83E-04	+	1.83E-04	+		
CF17_Dim30	6.78E-01	~	6.23E-01	~	1.04E-01	~	1.40E-02	+	2.73E-01	~	3.45E-01	~	2.57E-02	+	1.83E-04	+		
CF17_Dim50	2.83E-03	–	1.40E-01	~	2.11E-02	+	2.46E-04	+	1.83E-04	+	2.12E-01	~	2.46E-04	+	1.83E-04	+		
CF18_Dim30	9.26E-04	+	6.39E-05	+	6.39E-05	+	6.39E-05	+	6.39E-05	+	6.39E-05	+	NaN	~	NaN	~		
CF18_Dim50	1.42E-03	+	6.39E-05	+	6.39E-05	+	6.29E-05	+	6.39E-05	+	6.39E-05	+	NaN	~	NaN	~		
CF19_Dim30	3.32E-05	+	6.39E-05	+	1.42E-03	+	6.39E-05	+	6.39E-05	+	6.39E-05	+	NaN	~	NaN	~		
CF19_Dim50	6.39E-05	+	6.39E-05	+	6.39E-05	+	6.20E-05	+	6.39E-05	+	6.39E-05	+	NaN	~	NaN	~		
CF20_Dim30	1.69E-02	+	6.39E-05	+	6.39E-05	+	6.39E-05	+	6.39E-05	+	6.39E-05	+	NaN	~	NaN	~		
CF20_Dim50	1.69E-02	+	6.39E-05	+	6.39E-05	+	6.39E-05	+	6.39E-05	+	6.39E-05	+	NaN	~	NaN	~		
(+)		**20**		**22**		**24**		**38**		**38**		**29**		**27**		**34**		
(-)		**13**		**9**		**9**		**0**		**0**		**1**		**2**		**0**		
(~)		**7**		**9**		**5**		**2**		**2**		**10**		**11**		**6**		

Nomenclature: (+) indicate the better performance of TLCO compared to algorithm mentioned, (-) indicate the worse performance of TLCO compared to algorithm mentioned, and (~) indicates no significant difference of performance between compared algorithm and TLCO.

6. Engineering design problems

This section illustrates the reliability of TLCO to solve real-world problems, i.e. well-known engineering design problems including tension/compression spring, pressure vessel design, welded beam design and speed reducer problem, and structural optimization design problems of a 72-bar space truss design. These problems are used to test the performance of TLCO when there are many constraint conditions. The dead penalty function approach is used to solve the conditional constraints. To solve these problems, a set of 50 particles and 5000 iterations are used with 50 independence runs to report the best solution. The obtained results are compared with several similar techniques published in the literature.

To consider constrained optimization problems, the penalty method is used to handle the constraint violations. A constrained optimization problem is usually written as a nonlinear optimization problem of the following form:

Min $f(x)$ $x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$.

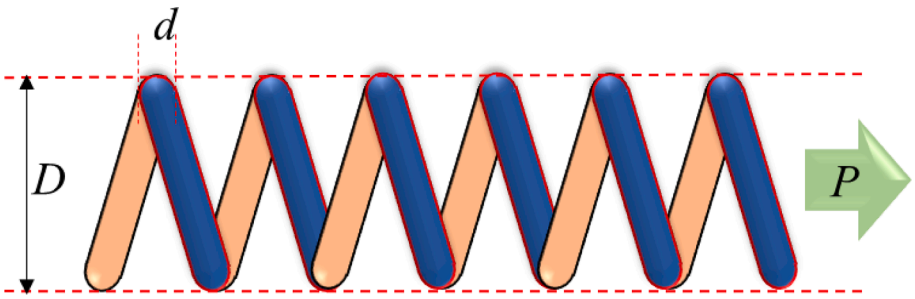

Fig. 17. Tension/compression spring design problem where the design variables are active coils (P), mean coil diameter (D), and wire diameter (d).

Subject to: $\begin{matrix} g_i(x) \leqslant 0 & i = 1, 2, ..., q \\ h_j(x) = 0 & j = q+1, q+2, ...m \end{matrix}$

The penalty function is given in Eq. (22)

$$\phi(x) = f(x) + \left[\sum_{i=1}^{q} r_i G_i + \sum_{j=q+1}^{m} c_i L_j \right] \tag{22}$$

where $\phi(x)$ is the new objective function to be optimized. r_i and c_j are penalty parameters G_i and L_j are the functions of $g_i(x)$ and$h_j(x)$, respectively. General formulas of G_i and L_j are given in Eq. (23)

$$\begin{matrix} G_i = \max[0,\ g_i(x)]^{\beta} \\ L_j = |h_j(x)|^{\gamma} \end{matrix} \tag{23}$$

where β and γ are commonly 1 or 2.

For engineering design problems mentioned in this section the condition $h_j(x) = 0$ is not considered. So, to solve these problems, the parameter $\beta = 2$ and $r_i = 10^{10}$ are selected to calculate the new objective function.

6.1. Tension/compression spring

Solving this problem requires to find the minimum weight of spring whose three changeable parameters d, D and P are wire diameter, mean coil diameter, and the number of active coils, respectively, as shown in Fig. 17.

The mathematical formulation of this problem is expressed as follows:

Give design variables:$x_1 = d$, $x_2 = D$, $x_3 = P$

Minimize:

$$f(x) = (x_3 + 2)x_2 x_1^2$$

Subject to:

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leqslant 0$$

$$g_2(x) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leqslant 0$$

$$g_3(x) = 1 - \frac{140.25 x_1}{x_2^2 x_3} \leqslant 0$$

$$g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leqslant 0$$

where

$$0.05 \leqslant x_1 \leqslant 2, \quad 0.25 \leqslant x_2 \leqslant 1.3, \quad 2 \leqslant x_3 \leqslant 15$$

Table 11 shows the best result obtained using TLCO with the corresponding values of constrained functions from $g_1(x)$ to$g_4(x)$. The results of other methods including 21 well-known optimization algorithms, and mathematical technique, which are published in the literature as shown in Table 12. CSA (Askarzadeh, 2016) gets the best performance with $f(x) = 0.01266523$. This value again was found by TLCO as $f(x) = 0.0126652328001663$. Especially, the error in constrained functions of TLCO can reach zero value for functions $g_1(x)$ and$g_2(x)$. It can be seen that the best optimal result using TLCO is very competitive in comparison with CSA and better than the other ones.

6.2. The problem of pressure vessel design optimization

The primary objective is to minimize the overall cost with four optimization variables including material, forming, and welding of a cylindrical vessel as shown in Fig. 18. There are four linear and nonlinear constraints affecting the design of pressure vessel. T_s and T_h are the thickness of shell and the thickness of the head, respectively. The inner radius is R and the length without considering the head is L.

Table 11
The best result obtained using TLCO for tension/compression spring design.

Variables	x_1	x_2	x_3	
	0.0516898670188459	0.356737128811723	11.2878291247229	
The value of constrained functions	g_1	g_2	g_3	g_4
	0	0	−4.04662730556043	−0.727715336112954
The best result $f(x)$	**0.0126652328001663**			

Table 12
Comparison of the best solution for welded beam design problem by different methods.

Different Algorithms	Optimum variables		Optimum weight	
	x_1	x_2	x_3	$f(x)$
PSO (He & Wang, 2007)	0.05172800	0.35764400	11.24454300	0.01267470
ES (Mezura-Montes & Coello, 2008)	0.355360	0.051643	11.397926	0.012698
GA (Coello, 2000)	0.05148000	0.35166100	11.63220100	0.01270480
GWO (Mirjalili, et al., 2014)	0.05169	0.356737	11.28885	0.0126660
BMO (Askarzadeh & Rezazadeh, 2013)	0.05165974	0.35601249	11.33044295	0.012665264
RO (A Kaveh & M Khayatazad, 2012)	0.051370	0.349096	11.762790	0.012679
CSA (Askarzadeh, 2016)	0.05168903	0.35671695	11.28901180	0.01266523
WOA (Mirjalili & Lewis, 2016)	0.051207	0.345215	12.004032	0.012676
DE (Huang, Wang, & He, 2007)	0.05160900	0.35471400	11.41083100	0.01267020
HS (Lee & Geem, 2005)	0.051154	0.349871	12.076432	0.012671
Constraint correction (Arora, 2004)	0.05000000	0.31590000	14.25000000	0.01283340
SSA (Mirjalili, et al., 2017)	0.051207	0.345215	12.004032	0.0126763
HHO (Heidari, et al., 2019)	0.051796393	0.359305355	11.138859	0.012665443
EO (Faramarzi, Heidarinejad, Stephens, & Mirjalili, 2020)	0.05161991	0.355054381	11.38796759	0.012666
MPA (Faramarzi, Heidarinejad, Mirjalili, & Gandomi, 2020)	0.051724477	0.35757003	11.2391955	0.012665
AVOA (Abdollahzadeh, Gharehchopogh, & Mirjalili, 2021)	0.051669833	0.356255347	11.316126	0.01266524
RLHHO (C. Li, Li, Chen, Jin, & Ren, 2021)	0.05172452	11.23910039	0.35757128	0.012665256
POA (Trojovský & Dehghani, 2022)	0.051892	0.361608	11.00793	0.012666
SO (Fatma A Hashim & Hussien, 2022)	0.0511	0.3418	12.2222	0.012672535

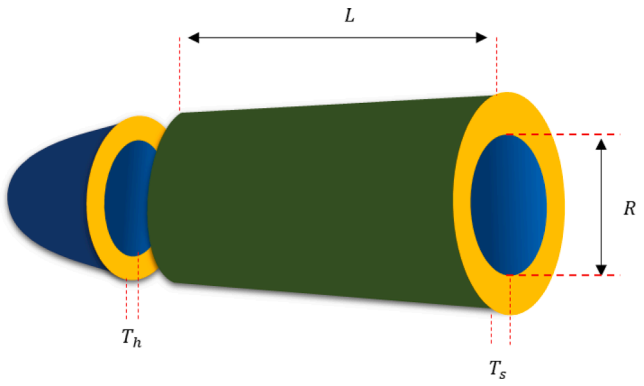

Fig. 18. Pressure vessel design problem where the design variables are inner radius R, the length L, thickness shell T_s and thickness of the head T_h.

The mathematical formulation of this problem is expressed as follows:

Give design variables$x_1 = T_s,\ x_2 = T_h,\ x_3 = R,\ x_4 = L$.

Minimize

$$f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

Subject to

$$g_1(x) = -x_1 + 0.0193x_3 \leqslant 0$$

$$g_2(x) = -x_2 + 0.00954x_3 \leqslant 0$$

$$g_3(x) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leqslant 0$$

$$g_4(x) = x_4 - 240 \leqslant 0$$

$$0 \leqslant x_1 \leqslant 99, \quad 0 \leqslant x_2 \leqslant 99, \quad 10 \leqslant x_3 \leqslant 200, \quad 10 \leqslant x_4 \leqslant 200$$

The best results gained by TLCO are listed in Table 13, while the results obtained by other optimization methods found in the literature are shown in Tables 14a and 14b.

Table 15, in the first case with a mixed variable problem, most of the algorithms find the best value. The best value in this case recorded belongs to the ABC algorithm with $f(x) = 6059.714339$. Here again TLCO appears to be the best algorithm in comparison with other algorithms with the best results reported for $f(x) = 6059.71433504844$. In case of continuous variable, the best value was registered $f(x) = 5885.33,277$ belong to EO algorithm. TLCO can reach this value with $f(x) = 5885.33277361646$ and it is the 1st rank in comparison with the rest of algorithms. The superiority of TLCO in solving this problem illustrates the efficiency of a new technique for movement updating in TLCO. TLCO can find a suitable movement by the proposed step length, which is short enough during the last few iterations to enhance the level of accuracy.

6.3. Welded beam design

The objective is to minimize the total cost of a welded beam as given in Fig. 19. There are four optimization variables including the thickness of weld (h), the length of weld (l), the thickness of bar (b) and the height of bar (t). This problem is designed with 7 constraint equations whose variables are shear stress(τ), bending stress in beam (θ), buckling load on bar (P_c) and the deflection (δ).

The mathematical formulation and the boundary constraints are provided as follows:

Given the design variables$x_1 = h,\ x_2 = l,\ x_3 = t,\ x_4 = b$.

Minimize:$f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$

Subject to:

$$g_1(x) = \tau(x) - \tau^{\max} \leqslant 0$$

$$g_2(x) = \sigma(x) - \sigma^{\max} \leqslant 0$$

$$g_3(x) = x_1 - x_4 \leqslant 0$$

$$g_4(x) = 1.1047x_1^2x_2 + 0.04811x_3x_4(14 + x_2) - 5 \leqslant 0$$

$$g_5(x) = 0.125 - x_1 \leqslant 0$$

$$g_6(x) = \delta(x) - \delta_{\max} \leqslant 0$$

$$g_7(x) = P - P_c(x) \leqslant 0$$

Where

Table 13
The best result obtained using TLCO for pressure vessel with mixed variable.

Variables	x_1	x_2	x_3	x_4
	0.8125	0.4375	42.0984455958549	176.636595842439
The value of constrained functions	g_1	g_2	g_3	g_4
	0	−0.0358808290155441	0	−63.3634041575606
The best result $f(x)$	**6059.71433504844**			

Table 14a
The best result obtained using TLCO for pressure vessel with continuous variable.

Variables	x_1	x_2	x_3	x_4
	0.778168641375106	0.384649162627902	40.3196187240987	200
The value of constrained functions	g_1	g_2	g_3	g_4
	−2.22044604925031e-16	0	0	−40
The best result $f(x)$	**5885.33277361646**			

Table 14b
The best result obtained using TLCO for pressure vessel with continuous variable.

Variables	x_1	x_2	x_3	x_4
	0.778168641375106	0.384649162627902	40.3196187240987	200
The value of constrained functions	g_1	g_2	g_3	g_4
	−2.22044604925031e-16	0	0	−40
The best result $f(x)$	**5885.33277361646**			

Table 15
Comparison of the best optimal results for pressure vessel design problem by different methods.

Different Algorithms	Optimum variables				Optimum cost
	x_1	x_2	x_3	x_4	$f(x)$
PSO (He & Wang, 2007)	0.812500	0.437500	42.091266	176.7465	6061.0777
ES (Mezura-Montes & Coello, 2008)	0.812500	0.437500	42.098087	176.640518	6059.745605
GA (Coello, 2000)	0.812500	0.437500	40.3239	200.000000	6288.7445
ABC (Akay & Karaboga, 2012)	0.812500	0.437500	42.098446	176.636596	6059.714339
CSA (Askarzadeh, 2016)	0.812500	0.437500	42.09844539	176.6365986	6059.714363
WOA (Mirjalili & Lewis, 2016)	0.812500	0.437500	42.0982699	176.638998	6059.7410
DE (Huang, et al., 2007)	0.812500	0.437500	42.098411	176.637690	6059.7340
ACO (A Kaveh & S Talatahari, 2010)	0.812500	0.437500	42.103624	176.572656	6059.0888
MVO (Mirjalili, Mirjalili, & Hatamlou, 2016)	0.8125	0.4375	42.090738	176.73869	6060.8066
EO (Faramarzi, Heidarinejad, Stephens, et al., 2020) Mixed variable	0.8125	0.4375	42.0984456	176.6365958	6059.7143
EO (Faramarzi, Heidarinejad, Stephens, et al., 2020) continuous variable	0.778168651	0.384649167	40.31961921	199.9999933	5885.33277
MPA (Faramarzi, Heidarinejad, Mirjalili, et al., 2020) Mixed variable	0.8125	0.4375	42.098445	176.636607	6059.7144
MPA (Faramarzi, Heidarinejad, Mirjalili, et al., 2020) Continuous variable	0.77816876	0.38464966	40.31962084	199.9999935	5885.3353
SMA (S. Li, Chen, Wang, Heidari, & Mirjalili, 2020)	0.7931	0.3932	40.6711	196.2178	5994.1857
HHO (Heidari, et al., 2019)	0.81758383	0.4072927	42.09174576	176.7196352	6000.46259
AVOA (Abdollahzadeh, et al., 2021)	0.778954	0.3850374	40.360312	199.434299	5886.676593
AOA (Abualigah, et al., 2021)	0.8303737	0.4162057	42.75127	169.3454	6048.7844
I-GWO (Nadimi-Shahraki, Taghian, & Mirjalili, 2021)	0.779031	0.385501	40.36313	199.4017	5888.34
SO (Fatma A Hashim & Hussien, 2022)	0.7819	0.3857	40.5752	196.5499	5887.529768

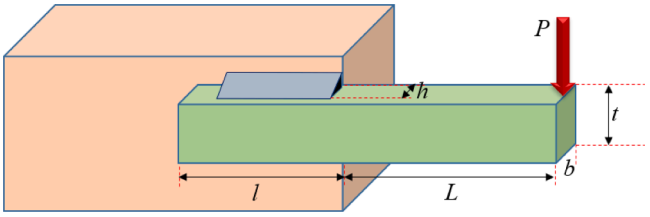

Fig. 19. Structure of Welded beam design. There are four optimization variables including the thickness of weld(h), the length of weld(l), the thickness of bar (b) and the height of bar(t).

Table 16
The best result obtained using TLCO for welded beam design.

Variables	x_1	x_2	x_3	x_4
	0.205729674316664	3.47048821435779	9.03662315198279	0.205729674316719
The value of constrained functions	g_1	g_2	g_3	g_4
	0	0	−5.50948175970234E-14	−3.27514756725735
	g_5	g_6	g_7	
	−0.0807296743166641	−0.235540321371264	−0.0026901544242719	
The best result $f(x)$	**1.72485243274265**			

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},\ \tau' = \frac{P}{\sqrt{2}x_1x_2},\ \tau'' = \frac{MR}{J},\ M = P\left(L + \frac{x_2}{2}\right)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},\ J = 2\left[\sqrt{2}x_1x_2\left\{\frac{x_2^2}{4}\right\} + \left(\frac{x_1 + x_3}{2}\right)^2\right],\ \sigma(x) = \frac{6PL}{x_4x_3^2},\ \delta(x) = \frac{4PL}{Ex_3^3x_4}$$

$$P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),\ P = 6000\ lb,\ L = 14\ in,\ E = 30 \times 10^6\ psi,\ G = 12 \times 10^6\ psi$$

$$\tau_{\max} = 13600\ psi,\ \sigma_{\max} = 30000\ psi,\ \delta_{\max} = 0.25\ in$$

Table 17
Comparison of the best solution for welded beam design problem by different methods.

Different Algorithms	Optimum variables				Optimum cost
	x_1	x_2	x_3	x_4	$f(x)$
ACO (A Kaveh & S Talatahari, 2010)	0.2057	3.471131	9.036683	0.205731	1.724918
GA (Coello & Montes, 2002)	0.205986	3.471328	9.020224	0.20648	1.728226
ES (Mezura-Montes & Coello, 2005)	0.20573	3.470489	9.036624	0.205729	1.724852
ABC (Akay & Karaboga, 2012)	0.20573	3.470489	9.036624	0.20573	1.724852
DE (Huang, et al., 2007)	0.203137	3.542998	9.033498	0.206179	1.733462
PSO (He & Wang, 2007)	0.202369	3.544214	9.04821	0.205723	1.728024
GWO (Mirjalili, et al., 2014)	0.205676	3.478377	9.03681	0.205778	1.72624
MVO (Mirjalili, et al., 2016)	0.205463	3.473193	9.044502	0.205695	1.72645
RO (A Kaveh & M Khayatazad, 2012)	0.203687	3.528467	9.004233	0.207241	1.735344
HS (Lee & Geem, 2005)	0.2442	6.2231	8.2915	0.2400	2.3807
DE (Huang, et al., 2007)	0.203137	3.542998	9.033498	0.206179	1.733462
EO (Faramarzi, Heidarinejad, Stephens, et al., 2020)	0.2057	3.4705	9.03664	0.2057	1.7249
HHO (Heidari, et al., 2019)	0.204039	3.531061	9.027463	0.206147	1.731991
SSA (Mirjalili, et al., 2017)	0.2057	3.4714	9.0366	0.2057	1.72491
SMA (S. Li, et al., 2020)	0.2054	3.2589	9.0384	0.2058	1.69604
MPA (Faramarzi, Heidarinejad, Mirjalili, et al., 2020)	0.205728	3.470509	9.036624	0.20573	1.724853
AVOA (Abdollahzadeh, et al., 2021)	0.20573	3.470474	9.036621	0.20573	1.724852
HBA (Fatma A. Hashim, et al., 2022)	0.2057	3.4704	9.0366	0.2057	1.72451
POA (Trojovský & Dehghani, 2022)	0.205719	3.470104	9.038353	0.205722	1.725021
SO (Fatma A Hashim & Hussien, 2022)	0.2057	3.4705	9.0366	0.2057	1.724852

$0.1 \leqslant x_1 \leqslant 2.0 \quad 0.1 \leqslant x_2, x_3 \leqslant 10 \quad 0.1 \leqslant x_4 \leqslant 2.0$

The best results obtained with TLCO for solving this problem are given in Table 16. And Table 17 shows the best values exploited by the other algorithms. It can be noticed that almost algorithms can find the best results with an acceptable error except HS (Lee & Geem, 2005). The best results in this problem are found to be around $f(x) = 1.72$. The best total cost exploited by TLCO is to reach this value with $f(x) = 1.72485243274265$ TLCO is rank 1st together with AVOA algorithm and SO algorithm. The error in constrained functions of TLCO can reach zero value for functions $g_1(x)$ and $g_2(x)$. *And* it can smaller 10^{-14} for function $g_3(x)$.

6.4. Speed reducer design problem

The objective is to minimize the total weight of a speed reducer while satisfying eleven constraints in total. This is considered as a more challenging benchmark because it has seven design variables as shown in Fig. 20. The variables of this problem include: the face with (x_1), the module of the teeth (x_2), the number of teeth on pinion (x_3), the length of the first shaft between bearings (x_4), the length of the first shaft between bearings (x_5), the diameter of the first shaft (x_6), and the diameter of the second shaft (x_7).

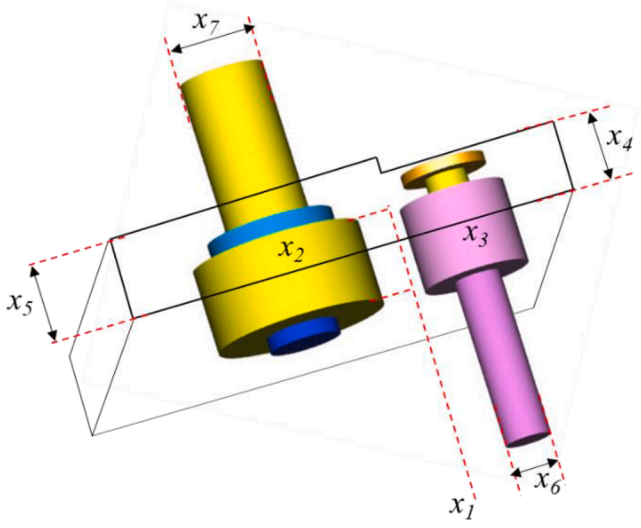

Fig. 20. A schematic of the speed reducer design where the design variables are the face with x_1, the module of the teeth x_2, the number of teeth on pinion x_3, the length of the first shaft between bearings x_4, length of the first shaft between bearings x_5, the diameter of the first shaft x_6, and the diameter of the second shaft x_7.

The mathematical formulation and the boundary constraints are provided as follows:

Minimize:

$$f(x) = 0.7854x_1x_2^2 \times \left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) - 1.508x_1\left(x_6^2 + x_7^2\right) + 7.4777\left(x_6^3 + x_7^3\right) + 0.7854\left(x_4x_6^2 + x_5x_7^2\right)$$

Subject to:

$$g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \leqslant 0 \quad g_3(x) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leqslant 0$$

$$g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leqslant 0 \quad g_4(x) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leqslant 0$$

$$g_5(x) = \frac{1}{110x_6^3}\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6} - 1 \leqslant 0$$

$$g_6(x) = \frac{1}{85x_7^3}\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6} - 1 \leqslant 0$$

$$g_7(x) = \frac{x_2x_3}{40} - 1 \leqslant 0 \quad g_8(x) = \frac{5x_2}{x_1} - 1 \leqslant 0$$

$$g_9(x) = \frac{x_2}{12x_2} - 1 \leqslant 0 \quad g_{10}(x) = \frac{1.5x_6 + 1.9}{x_5} - 1 \leqslant 0 \quad g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leqslant 0$$

$$2.6 \leqslant x_1 \leqslant 3.6, \quad 0.7 \leqslant x_2 \leqslant 0.8, \quad 17 \leqslant x_3 \leqslant 28, \quad 7.3 \leqslant x_4, x_5 \leqslant 8.3 \quad 2.9 \leqslant x_6 \leqslant 3.9, \quad 5 \leqslant x_7 \leqslant 5.5$$

Table 18 and Table 19, respectively, show the best result achieved by TLCO and the other algorithms. According to the statistical figures for this problem, PSO-DE algorithm has the best performance with $f(x) = 2996.348100$, while the best results of the other algorithms fluctuated between 2996 and 3010. FA algorithm fails to solve this problem when there is a big difference in comparison with the other algorithms. It can be seen that TLCO emerges as a unique algorithm that can provide a new

Table 18
Best solution obtained using TLCO for speed reducer design.

Variables	x_1 3.50000083096556	x_2 0.700000035409349	x_3 17.0000010315649	
	x_4 7.30008593690753	x_5 7.71534157591716	x_6 3.35021549742642	x_7 5.28665468872827
The value of constrained functions	g_1 −0.0739156501543081	g_2 −0.197998896021515	g_3 −0.499155113242193	
	g_4 −0.904643128033835	g_5 −6.02069843758102E-07	g_6 −1.22980777450543E-07	g_7 −0.70249996689864
	g_8 −1.86833902127148E-07	g_9 −0.58333325548586	g_{10} −0.0513367505542889	g_{11} −2.77606841614375E-06
The best result $f(x)$	**2994.47331784953**			

Table 19
Comparison of the best solution for speed reducer design problem by different methods.

Different Algorithms	Best solution							Optimum weight
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$f(x)$
AAO (Czerniak, Zarzycki, & Ewald, 2017)	3.499000	0.699900	17.000000	7.300000	7.800000	3.350200	5.287200	2996.783000
GWO (Mirjalili, et al., 2014)	3.501000	0.700000	17.000000	7.300000	7.811013	3.350704	5.287411	2997.819650
CS (X.-S. Yang & Deb, 2009)	3.501500	0.700000	17.000000	7.605000	7.818100	3.352000	5.287500	3000.981000
WSA (Baykasoğlu & Akpinar, 2017)	3.500000	0.700000	17.000000	7.300000	7.800000	3.350215	5.286683	2996.348225
MFO (Mirjalili, 2015b)	3.497455	0.700000	17.000000	7.827750	7.712457	3.351787	5.286352	2998.940830
SCA (Mirjalili, 2016b)	3.521000	0.700000	17.000000	8.300000	7.923351	3.355911	5.300734	3026.837720
AOA (Abualigah, et al., 2021)	3.503840	0.700000	17.000000	7.300000	7.729330	3.356490	5.286700	2997.915700
LGSI4 (Baykasoğlu & Akpinar, 2017)	3.501000	0.700000	17.000000	7.300000	7.800000	3.350214	5.286683	2996.348205
PSO-DE (Liu, Cai, & Wang, 2010)	3.500000	0.700000	17.000000	7.300000	7.800000	3.350210	5.286680	2996.348100
LGSI2 (Baykasoğlu & Akpinar, 2017)	3.500000	0.700000	17.000000	7.300000	7.800000	3.350215	5.286683	2996.348166
FA (Baykasoğlu & Ozsoydan, 2015)	3.507495	0.700100	17.000000	7.719674	8.080854	3.351512	5.287051	3010.137492
SO (Fatma A Hashim & Hussien, 2022)	3.4976	0.7	17	7.3	7.8	3.3501	5.2857	2995.54244
HBA (Fatma A. Hashim, et al., 2022)	3.4976	0.7	17	7.3	7.8	3.3501	5.2857	2995.4
HHO (Fatma A Hashim & Hussien, 2022)	3.4981	0.7	17	7.6398	7.8	3.3582	5.2853	3000.67208
POA (Trojovský & Dehghani, 2022)	3.5	0.7	17	7.3	7.8	3.350215	5.286683	2996.3482

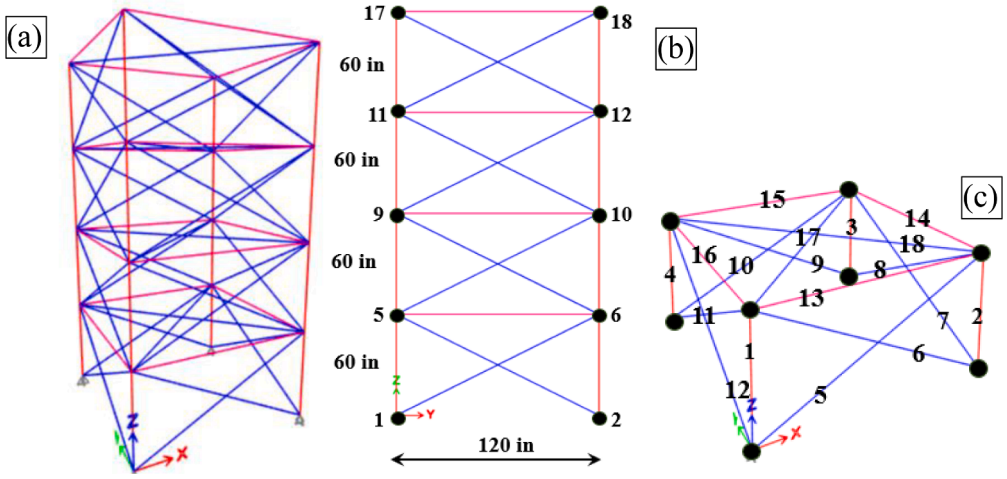

Fig. 21. 72-bar space truss design problem: (a) 3D model, (b) dimension and node, (c) element numbering pattern for first story.

Table 20
Load cases distribution on different nodes for the 72-bar space truss structure.

Nodes	Load case 1 (kips)			Load case 2 (kips)		
	P_x	P_y	P_z	P_x	P_y	P_z
17	5.0	5.0	−5.0	0.0	0.0	−5.0
18	0.0	0.0	0.0	0.0	0.0	−5.0
19	0.0	0.0	0.0	0.0	0.0	−5.0
20	0.0	0.0	0.0	0.0	0.0	−5.0

Table 21
Comparison between TLCO and optimization methods in the literature for 72-bar space truss design.

Groups	Members	TLCO	GGP (Adeli & Kamal, 1986)	GA (Cao, 1996)	ACO (Camp & Bichon, 2004)	PSO (Perez & Behdinan, 2007)	BB-BC (Camp, 2007)	HBB-BC (A. Kaveh & Talatahari, 2009)	CBO (A Kaveh & Mahdavi, 2015)	HHO (Al-Bazoon, 2021)	IMBA (Gholizadeh & Poorhoseini, 2016)
1	1–4	**1.9041**	2.0259	1.8562	1.9480	1.7427	1.8577	1.9042	1.920198	1.99	1.990
2	5–12	**0.5064**	0.5332	0.4933	0.5080	0.5185	0.5059	0.5162	0.506371	0.563	0.442
3	13–16	**0.1000**	0.1000	0.1000	0.1010	0.1000	0.1000	0.1000	0.1	0.111	0.111
4	17,18	**0.1000**	0.1000	0.1000	0.1020	0.1000	0.1000	0.1000	0.1	0.111	0.111
5	19–22	**1.2716**	1.1567	1.2830	1.3030	1.3079	1.2476	1.2582	1.225073	1.228	1.228
6	23–30	**0.5078**	0.5689	0.5028	0.5110	0.5193	0.5269	0.5035	0.515189	0.442	0.563
7	31–34	**0.1000**	0.1000	0.1000	0.1010	0.1000	0.1000	0.1000	0.1	0.111	0.111
8	35,36	**0.1000**	0.1000	0.1000	0.1000	0.1000	0.1012	0.1000	0.1	0.111	0.111
9	37–40	**0.5269**	0.5137	0.5177	0.5610	0.5142	0.5209	0.5178	0.561165	0.563	0.563
10	41–48	**0.5203**	0.4791	0.5227	0.4920	0.5464	0.5172	0.5214	0.519892	0.563	0.563
11	49–52	**0.1000**	0.1000	0.1000	0.1000	0.1000	0.1004	0.1000	0.1	0.111	0.111
12	53,54	**0.1000**	0.1000	0.1049	0.1070	0.1095	0.1005	0.1007	0.1	0.111	0.111
13	55–58	**0.1564**	0.1579	0.1557	0.1560	0.1615	0.1565	0.1566	0.156263	0.196	0.196
14	59–66	**0.5436**	0.5501	0.5501	0.5500	0.5092	0.5507	0.5421	0.53578	0.563	0.563
15	67–70	**0.4202**	0.3449	0.3981	0.3900	0.4967	0.3922	0.4132	0.435053	0.391	0.391
16	71,72	**0.5660**	0.4984	0.6749	0.5920	0.5619	0.5922	0.5756	0.550668	0.563	0.563
Weight (lb)		**379.6418**	379.31	380.32	380.24	381.91	379.85	379.66	379.812	389.334	389.33

level of accuracy. The best value exploited by TLCO straight ahead of all the other algorithms and sets a new performance record for solving this problem. The error in constraints of TLCO can be as smaller 10^{-7} as given in condition of $g_6(x),\ g_8(x),\ g_{11}(x)$ as shown in Table 18. This proves that TLCO can achieve a flexible movement, which is small enough during the last few iterations to exploit new search spaces, which current algorithms cannot approach because their movement strategy is not yet perfect.

6.5. Continuous 72-bar space truss design problem

The last engineering problem involved the 72-bar space truss structure, which is shown in Fig. 21. The truss has 72 bars and 20 nodes. The primary objective is to minimize the total weight of this structure. The model of this structure is implemented in MATLAB using one-dimensional finite element. During each iteration of TLCO, all stresses in 72 bars as well as all displacements at the 20 nodes are calculated by the finite element method (FEM). These results are then transmitted to TLCO to update the objective function. This process will continue until the best value is founded with the acceptable or reaching the number of desired iterations. The cross-sectional areas are classified into 16 groups, having the following values:

Group 1: $A_1 - A_4$; Group 2: $A_5 - A_{12}$; Group 3: $A_{13} - A_{16}$; Group 4: $A_{17} - A_{18}$; Group 5: $A_{19} - A_{22}$; Group 6: $A_{23} - A_{30}$; Group 7: $A_{31} - A_{34}$; Group 8: $A_{35} - A_{36}$; Group 9: $A_{37} - A_{40}$; Group 10: $A_{41} - A_{48}$; Group 11: $A_{49} - A_{52}$; Group 12: $A_{53} - A_{54}$; Group 13: $A_{55} - A_{58}$; Group 14: $A_{59} - A_{66}$; Group 15: $A_{67} - A_{70}$; Group 16: $A_{71} - A_{72}$;

The mathematical formulation and the boundary constraints are provided as follows:

Minimize: $f(x) = \sum_{i=1}^{72} \rho_i l_i A_i$

Subject to:

$$\sigma_{\min} \leqslant \sigma_i \leqslant \sigma_{\max},\ i = 1, 2, \ldots, 72$$

$$\delta_{\min} \leqslant \delta_i \leqslant \delta_{\max},\ i = 1, 2, \ldots, 72$$

where

$\rho = 0.1\ \frac{lb}{in^3}, E = 10^4 ksi$, $\sigma_{\min} = -25000\ ksi$, $\sigma_{\max} = 25{,}000\ ksi$, $\delta_{\min} = -0.25\ in$, $\delta_{\max} = 0.25\ in$,

Two cases of load distribution on different nodes for the 72-bar space truss structure are listed as shown in Table 20.

Table 21 shows that GGP gets the best performance with $f(x) = 379.31$. TLCO is the rank 2nd with $f(x) = 379.6418$ following HBB-BC algorithm with $f(x) = 379.66$. Although TLCO fails to achieve the best results with this problem, it can be seen that its accuracy level is only slightly lower than that of GGP and still performs better than other algorithms. Especially, TLCO demonstrates its superiority compared to the rest of algorithms.

7. Conclusion

This paper presented a novel metaheuristic optimization algorithm based on the concept of the life cycle of a termite colony and modulation of movement strategy. TLCO proposes a parallel structure for finding the best global optimization. This has been achieved through the specific task assignments of termite workers, soldiers and reproductive termites. Especially, reproductive termites will appear to replace termite soldiers and workers, when they get too much waste in a poor search exploited. Besides, the process of self-improvement of the initial population instead of each agent as the other algorithms is the novelty in TLCO. This will be a basic for calculating the timing of reproductive termites' emergence.

The step length S plays a key factor to establish the space of exploration or the space of exploitation. By introducing a linear function to control the step length S, the movement strategy in TLCO always ensures two important properties: (1) reaching a long movement during the first few iterations to improve the convergence rate and (2) reaching a short movement in later iterations to enhance the level of accuracy. Moreover, in order to make the movement more flexible, TLCO makes use of parameters to create two random movement directions. These parameters are ranged in the band [-1, 1] and can randomly receive negative or positive values during the process of position updating. If the parameters have negative values, the next position will move forward to potential search space around the current best solution, otherwise, the next position will move far away from the best solution. This will improve the exploration or exploitation in each termite worker and soldier.

Through various numerical examples, we have shown that TLCO performs better in comparison with other algorithms for a wide range of optimization problems and when applied to real engineering design problems. Especially, in high-dimension search space, TLCO still shows its power to find the best value with the smallest acceptable error, while other algorithms fail to find global optimum due to the local optima

problem. The ability of TLCO to escape local optima is also demonstrated by solving highly complex functions in CEC2005, whose many local optimal are presented. With large enough iterations, TLCO can still find the best value. Especially, for speed reducer design problem, the results reported using TLCO can be considered as a new record in this field. And for the remaining engineering problems, TLCO still reaches values that are close to the best optimal results obtained by other algorithms.

In conclusion, TLCO has proven to be highly reliable in solving optimization problems. It can be considered as a robust algorithm for solving real problems in many fields.

Declaration of Competing Interest

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

Data availability

No data was used for the research described in the article.

Acknowledgements

The authors acknowledge the financial support of VLIR-UOS TEAM Project, VN2018TEA479A103, 'Damage assessment tools for Structural Health Monitoring of Vietnamese infrastructures', funded by the Flemish Government. The authors wish to express their gratitude to Van Lang University, Vietnam, for financial support for this research.

References

Abdollahzadeh, B., Gharehchopogh, F. S., & Mirjalili, S. (2021). African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Computers & Industrial Engineering, 158*, Article 107408.

Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., & Gandomi, A. H. (2021). The arithmetic optimization algorithm. *Computer Methods in Applied Mechanics and Engineering, 376*, Article 113609.

Adeli, H., & Kamal, O. (1986). Efficient optimization of space trusses. *Computers & structures, 24*, 501–511.

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

Al-Bazoon, M. (2021). Harris Hawks optimization for optimum design of truss structures with discrete variables. *International Journal of Mathematical, Engineering and Management Sciences, 6*, 1157–1173.

Arora, J. S. (2004). *Introduction to optimum design*. Elsevier.

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

Askarzadeh, A., & Rezazadeh, A. (2013). A new heuristic optimization algorithm for modeling of proton exchange membrane fuel cell: Bird mating optimizer. *International Journal of Energy Research, 37*, 1196–1204.

Basturk, B. (2006). An artificial bee colony (ABC) algorithm for numeric function optimization. In *IEEE Swarm Intelligence Symposium, Indianapolis, IN, USA, 2006*.

Baykasoğlu, A., & Akpinar, Ş. (2017). Weighted Superposition Attraction (WSA): A swarm intelligence algorithm for optimization problems – Part 1: Unconstrained optimization. *Applied Soft Computing, 56*, 520–540.

Baykasoğlu, A., & Ozsoydan, F. B. (2015). Adaptive firefly algorithm with chaos for mechanical design optimization problems. *Applied Soft Computing, 36*, 152–164.

Benhamou, S. (2007). How many animals really do the Lévy walk? *Ecology, 88*, 1962–1969.

Beyer, H.-G., & Schwefel, H.-P. (2002). Evolution strategies–a comprehensive introduction. *Natural Computing, 1*, 3–52.

Biswas, A., Mishra, K., Tiwari, S., & Misra, A. (2013). Physics-inspired optimization algorithms: a survey. *Journal of Optimization, 2013*.

Biyanto, T. R., Matradji, I., Febrianto, H. Y., Afdanny, N., Rahman, A. H., Gunawan, K. S., Pratama, J. A. D., & Bethiana, T. N. (2017). Killer whale algorithm: An algorithm inspired by the life of killer whale. *Procedia Computer Science, 124*, 151–157.

Bonabeau, E., Marco, D., Dorigo, M., Théraulaz, G., & Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial systems*. Oxford University Press.

Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences, 237*, 82–117.

Camazine, S., Deneubourg, J.-L., Franks, N. R., Sneyd, J., Theraula, G., & Bonabeau, E. (2020). *Self-organization in biological systems*. Princeton University Press.

Camp, C. V. (2007). Design of space trusses using Big Bang-Big Crunch optimization. *Journal of Structural Engineering, 133*, 999–1008.

Camp, C. V., & Bichon, B. J. (2004). Design of space trusses using ant colony optimization. *Journal of Structural Engineering, 130*, 741–751.

Cao, G. (1996). *Optimized design of framed structures using a genetic algorithm*. The University of Memphis.

Chakraborty, A., & Kar, A. K. (2017). Swarm intelligence: A review of algorithms. *Nature-Inspired Computing and Optimization*, 475–494.

Chu, S.-C., Tsai, P.-W., & Pan, J.-S. (2006). Cat swarm optimization. In *Pacific Rim international conference on artificial intelligence* (pp. 854–858). Springer.

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

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

Cuong-Le, T., Minh, H.-L., Khatir, S., Wahab, M. A., Tran, M. T., & Mirjalili, S. (2021). A novel version of Cuckoo search algorithm for solving optimization problems. *Expert Systems with Applications, 186*, Article 115669.

Czerniak, J. M., Zarzycki, H., & Ewald, D. (2017). AAO as a new strategy in modeling and simulation of constructional problems optimization. *Simulation Modelling Practice and Theory, 76*, 22–33.

De Falco, I., Della Cioppa, A., Maisto, D., Scafuri, U., & Tarantino, E. (2012). Biological invasion–inspired migration in distributed evolutionary algorithms. *Information Sciences, 207*, 50–65.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B (Cybernetics), 26*, 29–41.

Droste, S., Jansen, T., & Wegener, I. (2006). Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems, 39*, 525–544.

Du, H., Wu, X., & Zhuang, J. (2006). Small-world optimization algorithm for function optimization. In *International conference on natural computation* (pp. 264–273). Springer.

Eusuff, M., Lansey, K., & Pasha, F. (2006). Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. *Engineering Optimization, 38*, 129–154.

Faramarzi, A., Heidarinejad, M., Mirjalili, S., & Gandomi, A. H. (2020). Marine Predators Algorithm: A nature-inspired metaheuristic. *Expert Systems with Applications, 152*, Article 113377.

Faramarzi, A., Heidarinejad, M., Stephens, B., & Mirjalili, S. (2020). Equilibrium optimizer: A novel optimization algorithm. *Knowledge-Based Systems, 191*, Article 105190.

Fister Jr, I., Yang, X.-S., Fister, I., Brest, J., & Fister, D. (2013). A brief review of nature-inspired algorithms for optimization. *arXiv preprint arXiv:1307.4186*.

Formato, R. (2007). Central force optimization: A new metaheuristic with applications in applied electromagnetics. *Prog Electromagn Res In., 77*, 425–491.

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

Garnier, S., Gautrais, J., & Theraulaz, G. (2007). The biological principles of swarm intelligence. *Swarm Intelligence, 1*, 3–31.

Gholizadeh, S., & Poorhoseini, H. (2016). Seismic layout optimization of steel braced frames by an improved dolphin echolocation algorithm. *Structural and Multidisciplinary Optimization, 54*, 1011–1029.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning.

Gong, D., Sun, J., & Ji, X. (2013). Evolutionary algorithms with preference polyhedron for interval multi-objective optimization problems. *Information Sciences, 233*, 141–161.

Gong, D., Sun, J., & Miao, Z. (2016). A set-based genetic algorithm for interval many-objective optimization problems. *IEEE Transactions on Evolutionary Computation, 22*, 47–60.

Grefenstette, J. J. (2013). *Genetic algorithms and their applications: proceedings of the second international conference on genetic algorithms*: Psychology Press.

Harifi, S., Khalilian, M., Mohammadzadeh, J., & Ebrahimnejad, S. (2019). Emperor Penguins Colony: A new metaheuristic algorithm for optimization. *Evolutionary Intelligence, 12*, 211–226.

Hashim, F. A., Houssein, E. H., Hussain, K., Mabrouk, M. S., & Al-Atabany, W. (2022). Honey Badger Algorithm: New metaheuristic algorithm for solving optimization problems. *Mathematics and Computers in Simulation, 192*, 84–110.

Hashim, F. A., & Hussien, A. G. (2022). Snake Optimizer: A novel meta-heuristic optimization algorithm. *Knowledge-Based Systems*, 108320.

Hatamlou, A. (2013). Black hole: A new heuristic optimization approach for data clustering. *Information Sciences, 222*, 175–184.

Haupt, R. L., & Ellen Haupt, S. (2004). Practical genetic algorithms.

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

Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., & Chen, H. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems, 97*, 849–872.

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

Hussain, K., Salleh, M. N. M., Cheng, S., & Shi, Y. (2019). Metaheuristic research: A comprehensive survey. *Artificial Intelligence Review, 52*, 2191–2233.

Juste, K., Kita, H., Tanaka, E., & Hasegawa, J. (1999). An evolutionary programming solution to the unit commitment problem. *IEEE Transactions on Power Systems, 14*, 1452–1459.

Kaveh, A., & Farhoudi, N. (2013). A new optimization method: Dolphin echolocation. *Advances in Engineering Software, 59*, 53–70.

Kaveh, A., & Khayatazad, M. (2012a). A new meta-heuristic method: Ray Optimization. *Computers & Structures, 112–113*, 283–294.

Kaveh, A., & Khayatazad, M. (2012b). A new meta-heuristic method: Ray optimization. *Computers & structures, 112*, 283–294.

Kaveh, A., & Mahdavi, V. (2015). Two-dimensional colliding bodies algorithm for optimal design of truss structures. *Advances in Engineering Software, 83*, 70–79.

Kaveh, A., & Talatahari, S. (2009). Size optimization of space trusses using Big Bang-Big Crunch algorithm. *Computers & structures, 87*, 1129–1140.

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

Kaveh, A., & Talatahari, S. (2010b). A novel heuristic optimization method: Charged system search. *Acta Mechanica, 213*, 267–289.

Keller, L. (1998). Queen lifespan and colony characteristics in ants and termites. *Insectes Sociaux, 45*, 235–246.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks* (Vol. 4, pp. 1942-1948 vol.1944).

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.

Korb, J., & Lenz, M. (2004). Reproductive decision-making in the termite, Cryptotermes secundus (Kalotermitidae), under variable food conditions. *Behavioral Ecology, 15*, 390–395.

Koza, J. R., & Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection* (Vol. 1). MIT Press.

Krause, J., Cordeiro, J., Parpinelli, R. S., & Lopes, H. S. (2013). A survey of swarm algorithms applied to discrete optimization problems. In *Swarm Intelligence and Bio-Inspired Computation* (pp. 169–191). Elsevier.

Krishnanand, K., & Ghose, D. (2009). Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intelligence, 3*, 87–124.

Le-Duc, T., Nguyen, Q.-H., & Nguyen-Xuan, H. (2020). Balancing composite motion optimization. *Information Sciences, 520*, 250–270.

Lee, K. S., & Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering, 194*, 3902–3933.

Lévy, P., & Lévy, P. (1954). *Théorie de l'addition des variables aléatoires*: Gauthier-Villars.

Li, C., Li, J., Chen, H., Jin, M., & Ren, H. (2021). Enhanced Harris hawks optimization with multi-strategy for global optimization tasks. *Expert Systems with Applications, 185*, Article 115499.

Li, S., Chen, H., Wang, M., Heidari, A. A., & Mirjalili, S. (2020). Slime mould algorithm: A new method for stochastic optimization. *Future Generation Computer Systems, 111*, 300–323.

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

Lu, X., & Zhou, Y. (2008). A novel global convergence algorithm: Bee collecting pollen algorithm. In *International Conference on Intelligent Computing* (pp. 518–525). Springer.

Mahdavi, S., Shiri, M. E., & Rahnamayan, S. (2015). Metaheuristics in large-scale global continues optimization: A survey. *Information Sciences, 295*, 407–428.

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

Mezura-Montes, E., & Coello, C. A. C. (2008). An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *International Journal of General Systems, 37*, 443–473.

MiarNaeimi, F., Azizyan, G., & Rashki, M. (2021). Horse herd optimization algorithm: A nature-inspired algorithm for high-dimensional optimization problems. *Knowledge-Based Systems, 213*, Article 106711.

Minh, H.-L., Khatir, S., Rao, R. V., Abdel Wahab, M., & Cuong-Le, T. (2021). A variable velocity strategy particle swarm optimization algorithm (VVS-PSO) for damage assessment in structures. *Engineering with Computers*.

Minh, H.-L., Khatir, S., Wahab, M. A., & Cuong-Le, T. (2021). An Enhancing Particle Swarm Optimization Algorithm (EHVPSO) for damage identification in 3D transmission tower. *Engineering Structures, 242*, Article 112412.

Minh, H.-L., Sang-To, T., Wahab, M. A., & Cuong-Le, T. (2022). A new metaheuristic optimization based on K-means clustering algorithm and its application for structural damage identification in a complex 3D concrete structure. *Knowledge-Based Systems*, 109189.

Mirjalili, S. (2015a). The ant lion optimizer. *Advances in Engineering Software, 83*, 80–98.

Mirjalili, S. (2015b). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228–249.

Mirjalili, S. (2016a). Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications, 27*, 1053–1073.

Mirjalili, S. (2016b). SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems, 96*, 120–133.

Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software, 114*, 163–191.

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

Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Computing and Applications, 27*, 495–513.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.

Moghaddam, F. F., Moghaddam, R. F., & Cheriet, M. (2012). Curved space optimization: a random search based on general relativity theory. *arXiv preprint arXiv:1208.2214*.

Moussaid, M., Garnier, S., Theraulaz, G., & Helbing, D. (2009). Collective information processing and pattern formation in swarms, flocks, and crowds. *Topics in Cognitive Science, 1*, 469–497.

Mucherino, A., & Seref, O. (2007). Monkey search: A novel metaheuristic search for global optimization. In *AIP conference proceedings* (Vol. 953, pp. 162–173). American Institute of Physics.

Mühlenbein, H., Gorges-Schleuter, M., & Krämer, O. (1988). Evolution algorithms in combinatorial optimization. *Parallel Computing, 7*, 65–85.

Nadimi-Shahraki, M. H., Taghian, S., & Mirjalili, S. (2021). An improved grey wolf optimizer for solving engineering problems. *Expert Systems with Applications, 166*, Article 113917.

Naruei, I., & Keynia, F. (2021). Wild horse optimizer: A new meta-heuristic algorithm for solving engineering optimization problems. *Engineering with Computers*, 1–32.

Noirot, C., & Pasteels, J. (1987). Ontogenetic development and evolution of the worker caste in termites. *Experientia, 43*, 851–860.

Oftadeh, R., Mahjoob, M., & Shariatpanahi, M. (2010). A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. *Computers & Mathematics with Applications, 60*, 2087–2098.

Pan, W.-T. (2012). A new fruit fly optimization algorithm: Taking the financial distress model as an example. *Knowledge-Based Systems, 26*, 69–74.

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine, 22*, 52–67.

Patnaik, S., Yang, X.-S., & Nakamatsu, K. (2017). *Nature-inspired computing and optimization* (Vol. 10). Springer.

Perez, R. E., & Behdinan, K. (2007). Particle swarm approach for structural design optimization. *Computers & Structures, 85*, 1579–1588.

Pham, D. T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S., & Zaidi, M. (2006). The bees algorithm—a novel tool for complex optimisation problems. In *Intelligent production machines and systems* (pp. 454–459). Elsevier.

Pinto, P. C., Runkler, T. A., & Sousa, J. M. (2007). Wasp swarm algorithm for dynamic MAX-SAT problems. In *International conference on adaptive and natural computing algorithms* (pp. 350–357). Springer.

Poorzahedy, H., & Rouhani, O. M. (2007). Hybrid meta-heuristic algorithms for solving network design problem. *European Journal of Operational Research, 182*, 578–596.

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A Gravitational Search Algorithm. *Information Sciences, 179*, 2232–2248.

Roth, M. (2005). Termite: A swarm intelligent routing algorithm for mobile wireless ad-hoc networks.

Sang-To, T., Hoang-Le, M., Khatir, S., Mirjalili, S., Wahab, M. A., & Cuong-Le, T. (2021). Forecasting of excavation problems for high-rise building in Vietnam using planet optimization algorithm. *Scientific Reports, 11*, 1–10.

Sang-To, T., Hoang-Le, M., Wahab, M. A., & Cuong-Le, T. (2022). An efficient Planet Optimization Algorithm for solving engineering problems. *Scientific Reports, 12*, 1–18.

Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper Optimisation Algorithm: Theory and application. *Advances in Engineering Software, 105*, 30–47.

Shah-Hosseini, H. (2011). Principal components analysis by the galaxy-based search algorithm: A novel metaheuristic for continuous optimisation. *International Journal of Computational Science and Engineering, 6*, 132–140.

Shiqin, Y., Jianjun, J., & Guangxing, Y. (2009). A dolphin partner optimization. In *In 2009 WRI global congress on intelligent systems* (Vol. 1, pp. 124–128). IEEE.

Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*, 702–713.

Storn, R., & Price, K. (1997). Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization, 11*, 341–359.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., & Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *KanGAL report*, 2005, 2005005.

Tang, K.-S., Man, K.-F., Kwong, S., & He, Q. (1996). Genetic algorithms and their applications. *IEEE Signal Processing Magazine, 13*, 22–37.

Thorne, B. L., Breisch, N. L., & Muscedere, M. L. (2003). Evolution of eusociality and the soldier caste in termites: Influence of intraspecific competition and accelerated inheritance. *Proceedings of the National Academy of Sciences, 100*, 12808–12813.

To, T. S., Le, M. H., Danh, T.-T., Khatir, S., Abdel Wahab, M., & Le, T. C. (2022). Combination of intermittent search strategy and an improve particle swarm optimization algorithm (IPSO) for model updating. *Frattura Ed Integrita Strutturale-Fracture And Structural Integrity, 16*, 141–152.

Trojovský, P., & Dehghani, M. (2022). Pelican optimization algorithm: A novel nature-inspired algorithm for engineering applications. *Sensors, 22*, 855.

Webster, B., & Bernhard, P. J. (2003). A local search optimization algorithm based on natural principles of gravitation. In.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*, 67–82.

Yang, C., Tu, X., & Chen, J. (2007). Algorithm of marriage in honey bees optimization based on the wolf pack search. In *The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007)* (pp. 462–467). IEEE.

Yang, X.-S. (2010a). Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-inspired Computation, 2*, 78–84.
Yang, X.-S. (2010b). A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)* (pp. 65–74). Springer.
Yang, X.-S., & Deb, S. (2009). Cuckoo search via Lévy flights. In *2009 World congress on nature & biologically inspired computing (NaBIC)* (pp. 210–214). IEEE.
Zervoudakis, K., & Tsafarakis, S. (2020). A mayfly optimization algorithm. *Computers & Industrial Engineering, 145*, Article 106559.
Zhao, W., Wang, L., & Zhang, Z. (2019). Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. *Knowledge-Based Systems, 163*, 283–304.