Termite life cycle optimizer

Hoang-Le Minh, Thanh Sang-To, Guy Theraulaz, Magd Abdel Wahab, Thanh Cuong-Le

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, Vietnam.

[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

*Corresponding author:
Dr. Thanh Cuong-Le (e-mail: cuong.lt@ou.edu.vn)

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.

Keyworks: CEC 2005, Optimal Engineering Design, Optimization, Termite Optimization Algorithm, Meta-heuristic, Stochastic optimization, Lévy flight.

1 INTRODUCTION

Optimization algorithms allow us to find solutions to optimization problems [1]. 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 $f(x_0) \leq f(x)$ for all $x \in A$ (minimization)

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

Besides using the approaches of mathematics and numerical analysis, meta-heuristic algorithm is considered an effective approach to solve optimization problems in various domains. Meta-heuristic is designed for solving a problem more quickly when traditional methods are too slow, or for searching the best solution with an acceptable error when classic methods fail to find the exact solution. This is achieved by the repeat the process of "trial and error" continuously, 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 [2]:

- 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.

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 solution, 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 suffer from early convergence, thus the accuracy level is limited. On the contrary, population-based algorithms randomly generate a set of solution candidates $\overrightarrow{X_i} = \left\{\overrightarrow{X_1}, \overrightarrow{X_2}, ...\overrightarrow{X_n}\right\}$ in search space at the first step. Then, these candidates are combined/updated to explore and exploit the new search space $\overrightarrow{X'_i} = \left\{\overrightarrow{X'_1}, \overrightarrow{X'_2}, ...\overrightarrow{X'_n}\right\}$ 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 the large search space [2-4]. 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 of the previous step because of the useful information recorded. Besides advantages, high computational cost and the need for more function evaluation are two major drawbacks of population-based algorithms.

Many metaheuristic algorithms with different inspiration can be divided into three classes [5] such as: evolution-inspired [6, 7], physics-inspired [8] and biological swarm-inspired [9]. Evolution-inspired method is population-based approach and is inspired by the laws of biological systems [10, 11]. The advantages of these algorithms are that each solution candidate is be 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 proposed by Holland [12] that simulates the Darwinian evolution. GA used techniques including mutation, crossover, to improve the solution candidate. The original version and its variants have widely applied to many real-world problems [13-15]. Other popular algorithms were presented including Evolution Strategy (ES) [16], Genetic Programming (GP) [17], Differential Evolution (DE) [18], Evolutionary Programming (EP) [19]. Biogeography-Based Optimization algorithm (BBO) [20].

Physics-inspired method is inspired by physical phenomena in nature and is population-based approach. Simulated annealing (SA) algorithm [21]. At each step iteration, SA registers some neighboring state s^* of the current state, and probabilistically decides between moving the system to state s^* or staying 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) [22], Gravitational Search Algorithm (GSA) [23], Charged System Search (CSS) [24], Small-World Optimization Algorithm (SWOA) [25], Central Force Optimization (CFO) [26], Galaxy-based Search Algorithm (GbSA) [27], Black Hole [28], Ray Optimization (RO) [29], curved space optimization (CSO) [30], Atom search optimization (ASO) [31] 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 [32, 33]. 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 [34-36]. 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. [37] 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 amount to 48% in total including 23%, 16% and 9%. The popular algorithms that belong to these percentages are listed in **Table 1**.

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

Algorithm	Year of proposed
1. Particle swarm optimization (PSO) [38]	1995
2. Ant colony optimization (ACO) [39]	1996
3. Bacterial foraging optimization (BFO)	2002
4. Artificial bee colony algorithm (ABC) [40]	2005
5. Termite Algorithm (TA) [41]	2005
6. Glowworm swarm optimization (GSO) [42]	2005
7. Shuffled frog leaping algorithm (SFLA) [43]	2006
8. Cat Swarm Optimization (CAT) [44]	2006
9. Bees algorithm (BA) [45]	2006
10. Wasp Swarm Algorithm (WSO) [46]	2007
11. Monkey search (MA) [47]	2007
12. Wolf pack search algorithm [48]	2007
13. Bee Collecting Pollen Algorithm (BCPA) [49]	2008
14. Cuckoo search (CS) [50]	2009
15. Dolphin Partner Optimization (DPO) [51]	2009
16. Bat algorithm (BA) [52]	2010
17. Firefly Algorithm (FA) [53]	2010
18. Hunting Search (HS) [54]	2010
19. Bird Mating Optimizer (BMO) [55]	2012
20. Krill Herd (KH) [56]	2012
21. Fruit fly Optimization Algorithm (FOA) [57]	2012
22. Dolphin Echolocation (DE) [58]	2013
23. The Smell Detection Agent (SDA) [58]	2014
24. Grey Wolf optimizer (GWO) [59]	2014
25. The ant lion optimizer (ALO) [60]	2015
26. Dragonfly algorithm [61]	2016
27. The Whale Optimization Algorithm (WOA) [62]	2016
28. Killer Whale Algorithm [63]	2016
29. Grasshopper optimization algorithm (GOA) [64]	2017
30. Salp Swarm Algorithm (SSA) [65]	2017
31. Emperor Penguins Colony (EPC) [66]	2019
32. A mayfly optimization algorithm (MA) [67]	2020
33. Jellyfish Search (JS) [68]	2021

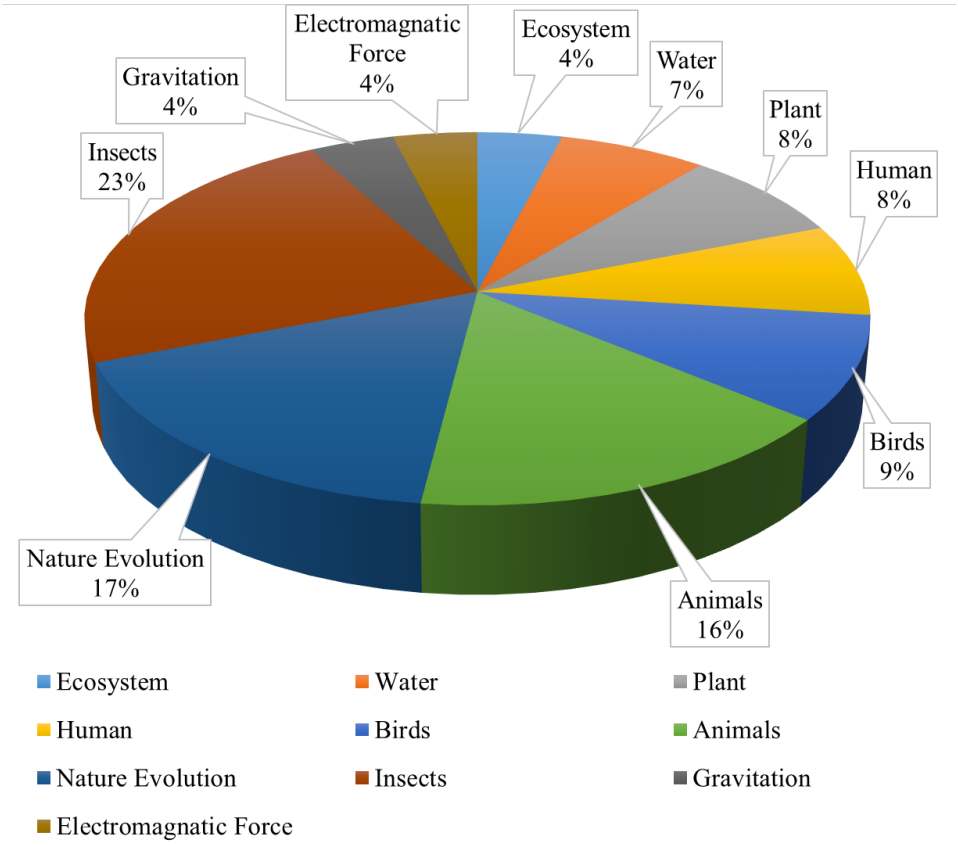


Fig. 1: Metaphors adopted by researchers for designing new metaheuristics

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 for the following reasons: (i) Simplicity is the primary advantage of Swarm-inspired algorithms, the majority of algorithms in this field follow a simple structure and have 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 [69]. This means that the process of derivation of the mathematical model is ignored. In another word, the optimization algorithms focus on changing the input and monitor the output for reaching the target (objective function).

Particle swarm optimization (PSO) [38] 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 [39], 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 iteration is sufficient.

The concept in PSO can be perceived as originality 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 the different approaches through simulating swarm intelligence by mathematical models. Grey Wolf optimizer (GWO) [59] 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 was register either larger than one or smaller than one. Between each iteration, the shared information was secured by the updating three best solutions in GWO. The specificity 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 supports GWO to achieve a good convergence rate and a high accuracy level. Firefly Algorithm (FA) [53] 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 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 was more than 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 comparison 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 the PSO [38]. 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) [23], Cuckoo search (CS) [50], Bacterial foraging optimization (BFO) [10] and so on.

The development of social technologies opens new challenges dealing with a great number of optimization problems. Although a large of number optimization algorithms have introduced in the literature, new optimization algorithms are still being developed to solve emerging complex optimization problems to obtain a better scheme. No Free Lunch Theorem of Optimization [70] proved that there is no optimization algorithm performing the best overall different types of problems. It means 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 field

In this paper, a novel Swarm-inspired algorithm is proposed for solving the optimization problem named termite life cycle optimizer (TLCO) based on the termites life cycle and the modulation of movement strategies in nature. To the best of our knowledge, there is no related research this study found 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 [71]. 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; they perform no other function. A queen can lay thousands of eggs each year. During the two-week incubation period, the termite worker take 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 charged 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 [72].

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 [73]. 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 [74].

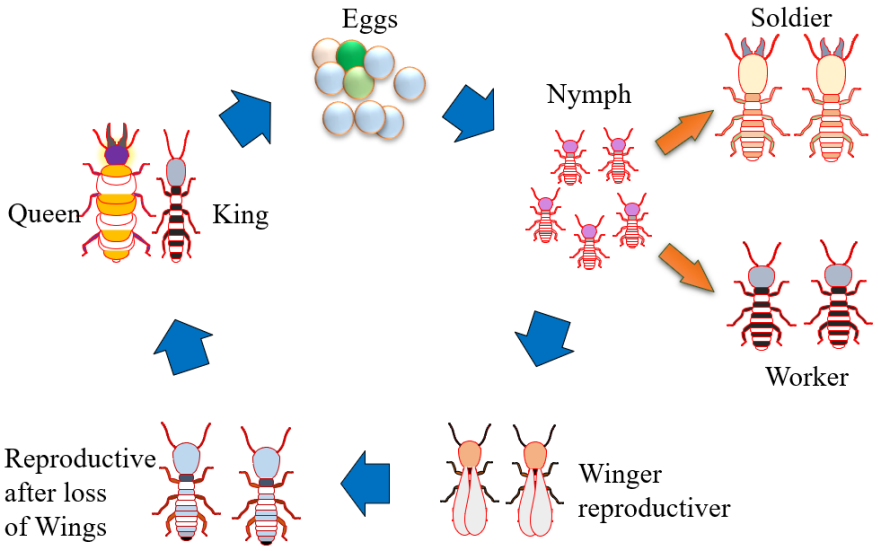


Fig. 2: Life cycle of a Termite colony

Communication between termites is considered to be crucial element for the coordination of individuals' activities and the emergence of collective intelligence [75]. This characteristic is a significant factor for development and survival of the colony. Most termites are blind, so communication primarily occurs 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.

The 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) guarantee the ability of exploration and exploitation of the algorithm, (ii) the ability to share information among each particle in swarm, (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**.

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 particle in swarm
Worker caste	Find the new food of source, build the shelter tubes	The particular particles which have the ability to explore the new search space
Soldier caste	Protect colony and attack the other intruders	The particular particles which 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 abandon the bad current solution and replace it with a new potential solution.
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.

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 [76] . 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 expressed in Eq. (1)

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

Where $S_i = (s_1, s_2, ..., s_D)$ denotes as 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 [77]. If each step $S_i = (s_1, s_2, ..., s_D)$ in the random walk obeys a Lévy distribution, the random walk will become a Lévy flight [50]. According to Mantegna's algorithm, the step length S_i is defined as Eq. (2)

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

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

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

$$U \sim N\left(0, \sigma_u^2\right), \quad V \sim N\left(0, \sigma_v^2\right) \tag{3}$$

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

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

In Eq. (4) the Gamma function Γ for an integer z is expressed as 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 a large search space and the exploitation near the best solution inn that space. These characteristics may provide some hints and insights into how and why metaheuristic algorithms behave.

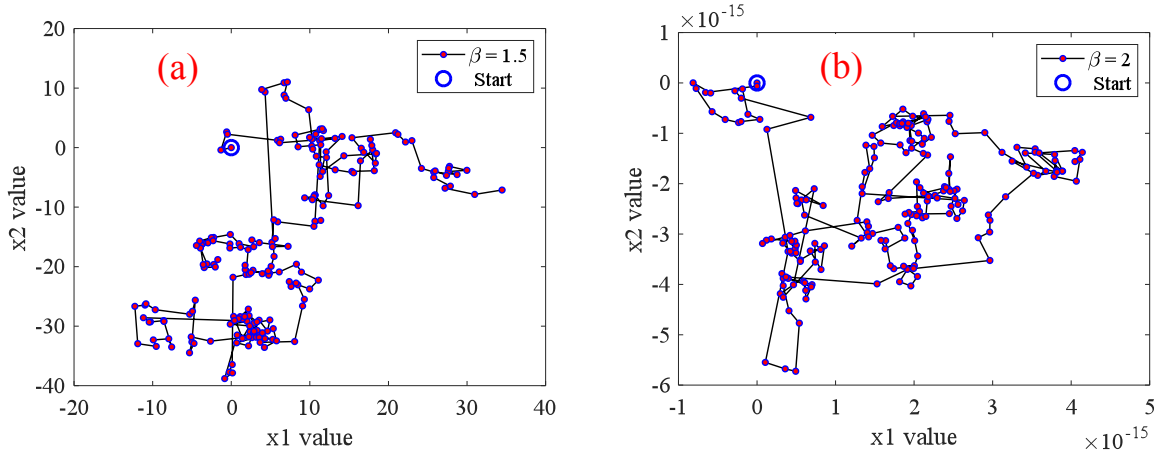


Fig. 3: Random walk in 2D dimension using Lévy flight in 200 steps length with different β: (a) simulation over long distance, (b) simulation over short distance

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 value range is bounded 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 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 few first iterations, the boundary condition of β value will be changed from range $[1, 2]$ to range $[1.5, 2]$ for improving convergence rate and given in Eq. (6) The effect of the values of β on step length at each iteration as shown in **Fig. 4**

$$\beta = 1.5 + \frac{0.5k}{K_{\max}} \qquad (6)$$

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

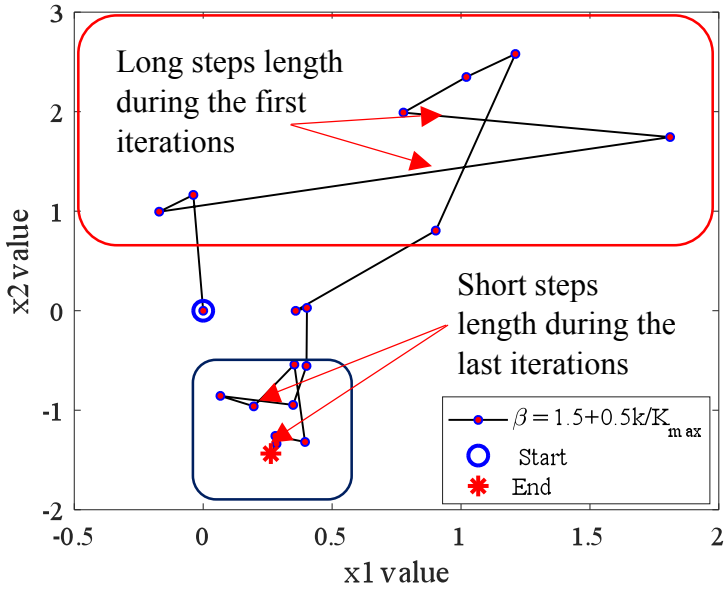


Fig. 4: Random walk in 2D dimension using Lévy flight in 20 steps length with proposed β

3.3 The movement strategy of termite workers in the model

In this paper, we assume that the number of termite workers accounts for 70 % of the total number of individual in the colony. Let's assume an initial population of termites in the colony of size N; thus, $X_{i,worker}, 1 \le i \le 0.7N$ denotes the position of termite workers. The primary duties of termite workers are to explore the source of food and build the shelter tubes. In the mathematical model, the process of position updating between k^{th} and $(k+1)^{th}$ iteration can be described as Eq. (7).

$$X_{i,worker}^{k+1} = X_{i,worker}^{k} + \theta_1 \left(dX_{i,worker}^{k+1} \right)$$
$$\theta_1 = -1 + 2rand(1) \qquad (7)$$

Where θ_1 is a scalar number whose value is limited in range $[-1,1]$. It uses to control the movement direction of $dX_{i,worker}^{k+1}$ and makes the movement of termite workers become more flexible. Thus, at the step $(k+1)^{th}$, $X_{i,worker}^{k+1}$ can randomly move in two directions, the first direction is secured if $\theta_1 < 0$ and the remaining direction if $\theta_1 > 0$.

$dX_{i,worker}^{k+1}$ is intelligent movement strategy of termite workers to explore the new search space at $(k+1)^{th}$ iteration and is expressed in Eq. (8).

$$dX_{i,worker}^{k+1} = dS(k+1) \otimes \left| X_{best}^{k} - X_{i,worker}^{k} \right|$$
$$dS(k+1) = \left(rand(1,D) + S(k+1) \right) \qquad (8)$$

Where $dS(k+1)$ is a scalar vector representing the change in the value of step length $S(k+1)$. X_{best}^{k} is the best solution recorded at k^{th} iteration. The symbol $\otimes$ is a point-to-point multiplication. The term $rand(1,D)$ is a scalar vector having its value within the range $[0,1]$ and having a dimension D. $S(k+1)$ is step length at iteration $(k+1)^{th}$ given in Eq. (2) whose β is calculated according to Eq. (6).

To simplify, Eq. (8) is rewritten as Eq. (9) and the position updating process of each termite worker as shown in **Fig. 5**.

$$X_{i,worker}^{k+1} = X_{i,worker}^{k} + \left(-1+2rand(1)\right)\left[\left(rand(1,D)+S(k+1)\right)\otimes\left|X_{best}^{k} - X_{i,worker}^{k}\right|\right] \quad (9)$$

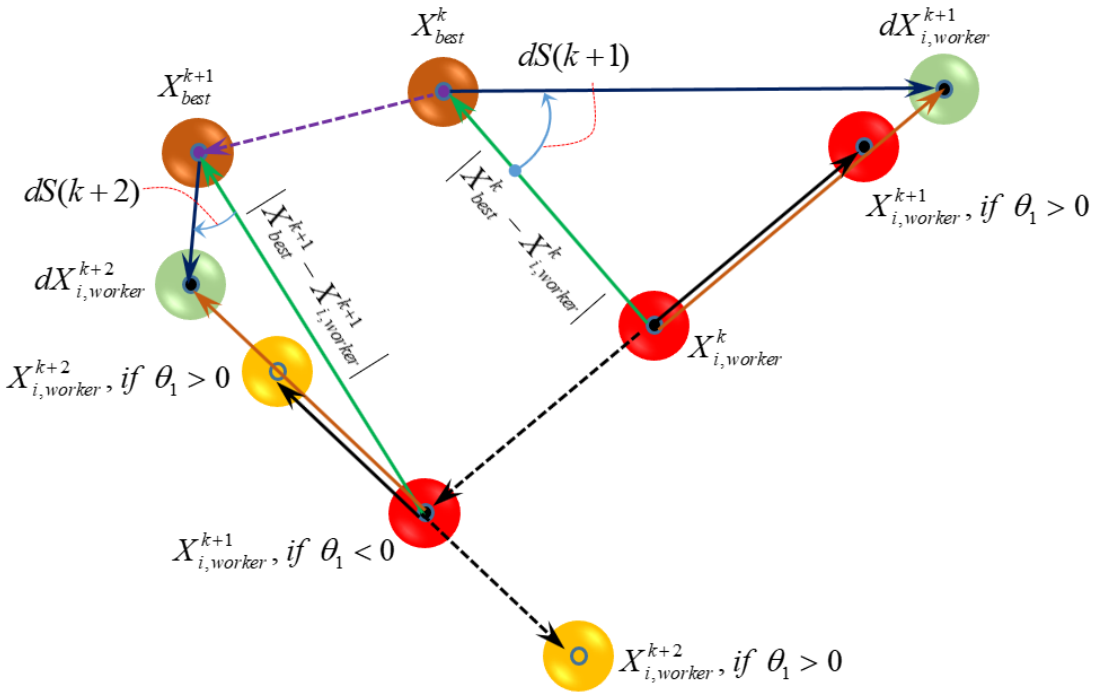


Fig. 5: The movement of termite workers at iterations $(k+1)^{th}$ and $(k+2)^{th}$

$dX_{i,worker}^{k+1}$ is a vector to ensure the ability to expand the search space of termite workers. It is produced by a combination of two vectors. The first vector is $\left|X_{best}^{k} - X_{i,worker}^{k}\right|$ that called a fixed-component, and the second vector is $\left(rand(1,D)+S(k+1)\right)$ which is called a random component $dS(k+1)$. It should be emphasized that the step length $S(k+1)$ will control the fluctuation of $dS(k+1)$, Thus, If $dS(k+1)$ creates a wide search space, it can induce many wasteful movements which is denoted as follows $f_{of\,un}(X_{i\ worker}^{k}) > f_{of\,un}(X_{,i\ worker}^{k+1})$, where $f_{of\,un}$ is the objective function. This will seriously affect the convergence rate of the algorithm. Otherwise, if $dS(k+1)$ creates a narrow search space, it will affect the accuracy level due to local optima problem. To overcome these problems, TLCO proposed a new step length S

controlled by the parameter β whose value increases from 1.5 to 2 over the course of iterations given in Eq. (6). As a result, the vector $\left(rand(1,D)+S(k+1)\right)$ can get either long enough (during the few first iterations) to improve the convergence rate or short enough (during a few last iterations) to improve the accuracy level. **Fig. 5** illustrates the updating of the position of each termite worker at iterations $(k+1)^{th}$ and $(k+2)^{th}$; because of the control of β, the individual value $s_i(i=1,2,...,D)$ of step length S will gradually decreases as the number of iterations increases. Thus, $dS(k+2)$ will have less fluctuation than $dS(k+1)$, which will improve termite worker's position over the course of iterations. Especially, at the final iteration $\left(k=K_{\max}\right)$, the step length S will approach an infinitesimal value and the direction movement of each termite worker will be the same of the vector $\left|X_{best}^{k}-X_{i,worker}^{k}\right|$ as shown in **Fig. 6**. Moreover, the movement direction of $X_{i,worker}^{k+1}$ is decided by the value of θ_1. If the value of θ_1 is negative $\left(\theta_1<0\right)$, $X_{i,worker}^{k+1}$ undergoes a trend move forward to the potential search space. By contrast, if the value of θ_1 is positive $\left(\theta_1>0\right)$, $X_{i,worker}^{k+1}$ will move far from its current position. Because of these constraints, this makes the movement strategy of termite workers more flexible.

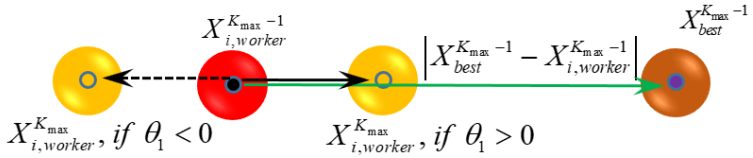


Fig. 6: The movement trend of workers termite at the final iteration $(k=K_{\max})$

3.4 The movement strategy of reproductive termites in the model

The primary task of reproductive termites is to create a new colony. In a mathematical model, this process can be achieved by evaluating the performance of the $X_{i,worker}$. If $X_{i,worker}$ performs wasteful movements too frequently over the course of iterations, the reproductive termites $X_{i,reproductive}$ will emerge to find the new potential search space.

In TLCO, the timing of their appearance is determined by a control parameter λ called Limit. Thus, after each iteration if $X_{i,worker}$ cannot explore a new better food source, *i.e.* if the condition $f_{of\ un}(X^{k}_{i\ worker})<f_{of\ un}(X^{k+1}_{,i\ worker})$ is not satisfied, these times will be recorded and counted through iterations by a pre-determined number called *Trial*. And if the condition $Trial\geq\lambda$ is satisfied, reproductive termites will appear to fly to a new potential region for establishing a new colony. This process is described in mathematical form as follows:

Set: $\lambda=\mu K_{max}$

Where μ is the parameter to specify when the reproductive termites will occur, and they can be adjusted by user for a particular structure. Especially, if $\mu = 1$, reproductive termites will be ignored in TLCO. The position of the new colony explored by each reproductive termite is expressed in Eq. (10)

$$\begin{aligned} &if\ Trial \geq \lambda \\ &X_{i,reproductive}^{k+1} = X_{best} + S(k+1) \end{aligned} \tag{10}$$

Based on the characteristics of the step length vector $S(k+1)$, the new position of each reproductive termite $X_{i,reproductive}^{k+1}$ can obtain both conditions; the first is far a way from X_{best} during a few first iterations and the second is close to X_{best} in a few last iterations as shown in **Fig. 7.** Especially, if μ is equal one, TLCO will ignore the reproductive termites phase.

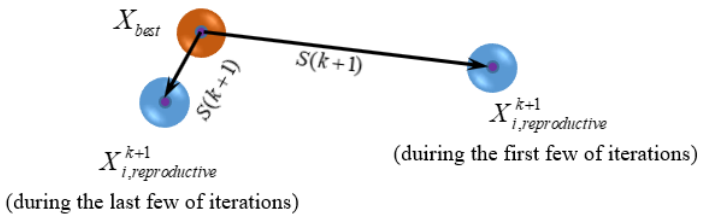


Fig. 7: The movement trend of reproductive termite at $(k+1)^{th}$ iteration

3.5 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 $X_{i,soldier}, 0.7N < i \leq N$ 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 X_{best}. In the model, the update of their position is expressed in Eq. (11)

$$X_{i,soldier}^{k+1} = X_{best} + dX_{best} \tag{11}$$

Where dX_{best} is the movement strategy of termite soldiers that controls their new positions. To match their mission, dX_{best} must be a vector to ensure the ability of exploitation of TLCO and is expressed in Eq. (12).

$$\begin{aligned} &dX_{best} = \theta_2 \left| X_{best} \otimes S(k+1) - X_{i,soldier}^{k} \right| \\ &\theta_2 = -1 + 2rand(1) \end{aligned} \tag{12}$$

The term $\left| X_{best} \otimes S(k+1) - X_{i,soldier}^{k} \right|$ where $\otimes$ is a point-to-point multiplication creates the trend of movement towards the high-density region defined by $X_{best} \otimes S(k+1)$. The term $S(k+1)$ is

added in Eq. (12) as the primary vector for exploiting the new search space around X_{best}. θ_2 is a parameter to adjust the attack direction of each soldier termite. Resembling to θ_1 mentioned in Eq. (7), it can have either negative or positive values.

To simplify, Eq. (12) is rewritten as Eq. (13).

$$X_{i,soldier}^{k+1} = X_{best}^{k} + \left(-1 + 2rand(1)\right)\left|X_{best} \otimes S(k+1) - X_{i,soldier}^{k}\right| \qquad (13)$$

Eq. (13) fully simulates the behavior of termite soldiers. Even if a new best solution is found at each iteration, this equation ensures that each termite soldier will always find a flexible way to move close the new best exploited solution. We assume that at k^{th} iteration, the best solution is X_{best}^{k} and the best solution is updated at $(k+1)^{th}$ iteration called X_{best}^{k+1}. The update of the position of each soldier termite at iterations $(k+1)^{th}$ and $(k+2)^{th}$ is illustrated in **Fig. 8**.

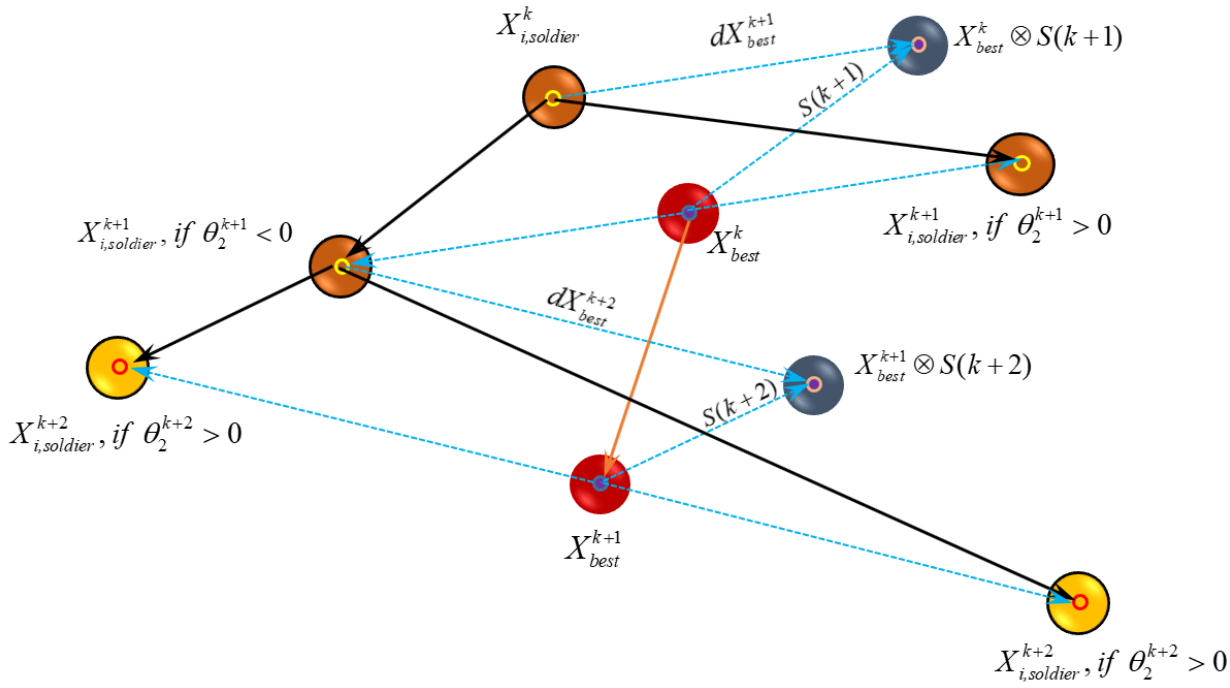


Fig. 8: The movement trend of soldier termite at iterations $(k+1)^{th}$ and $(k+2)^{th}$

3.6 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:

- 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 few first iterations so as to improve the convergence rate and (2) enough short distance during the a few last iterations to enhance the level of accuracy.

- 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% are devoted to exploration and 30% to exploitation of termite soldiers.
- The space of exploration or exploitation is controlled by the value of the step length S and two parameters θ_1 and θ_2 whose values are in the range $[-1,1]$. Based on these parameters, the movement direction in TLCO becomes more flexible to escape from local optima.
- TLCO allows the user to decide when the reproductive termites will emerge through the value of the λ parameter. Note that if μ equal one, TLCO will ignore the reproductive termites phase.

The proposed TLCO algorithm is outlined below:

Algorithm 1: The implementation process of TLCO to find the best solution

1. Initialize the termite population X_i $(i = 1,2,...,N)$;
2. Calculate the objective function of each termite $f(X_i), i = 1,2,...,N$;
3. Update the best solution X_{best} and the best objective function $f(X_{best})$;
4. Initialize the value of the *Limit* λ;
5. **For** k=1: K_{max}
6. Calculate the parameter β at each iteration using Eq. (6);
7. Calculate the step length S at each iteration using Eq. (2);
8. ----------
9. ***% % Start the tasks of termite workers and reproductive termites***
10. **For** each termite worker X_i $(1 \le i < 0.7N)$
11. Update the position of each termite worker using Eq. (7);
12. Calculate the objective function of each termite work $f(X_i), i = 1,2,...,0.7N$;
13. **If** $f(X_i), (i = 1,2,...,0.7N) < f(X_{best})$;
14. Update the best solution X_{best} and the best objective function $f(X_{best})$;
15. **Else**
16. *Trial(i)* = *Trial(i)* +1;
17. **End If**
18. **If** *Trial(i)* ≥ *Limit*
19. Reproductive termites will occur according to Eq. (10);
20. Calculate the objective function of reproductive termite;
21. Update the best solution X_{best} and the best objective function $f(X_{best})$;

```
            Trial(i) = 0;
        End If
    End For
% % Stop the tasks of termite workers and reproductive termites
----------------------------------------------------------------------------------------------------
% %  Start the task of termite soldiers
    For each termite soldier X_i (0.7N ≤ i ≤ N);
        Update the position of each termite soldier using Eq. (11);
        Calculate the objective function of each termite soldier f(X_i), i = 0.7N,...,N;
        Update the best solution X_best and the best objective function f(X_best);
    End For
% % Stop the task of termite soldiers
----------------------------------------------------------------------------------------------------
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 [59, 78, 79]. The first family test functions (F1-F7) shown in **Table 3** has 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) 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, search history of the 1st termite termite worker $\left(X_{i,worker}, i=1\right)$, the 1st reproductive termite in case it happens $\left(X_{1,reproductive}\right)$, and the 1st termite soldier $\left(X_{i,soildier}, i=0.7N\right)$ in 2D dimension, and trajectory curve of the first two variables $\left(x_1, x_2\right)$ of the best solution $X_{best}=\left(x_1, x_2, ..., x_D\right)$ found at each iteration, whose results show in **Fig. 8** with some typical functions (F1, F4, F6, F7, F9, F10, F14, F17, F19, F23). TLCO uses a number of termites $N = 30$ during 1000 iterations. The number of termites is

divided in two groups 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.

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
$F_4(x)=\max\left\{\lvert x_i\rvert, 1\le i\le n\right\}$	$[-100,100]^D$	0
$F_5(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)+\left(x_i-1\right)^2\right]$	$[-30,30]^D$	0
$F_6(x)=\sum_{i=1}^{n}\left(\left[x_i+0.5\right]\right)^2$	$[-100,100]^D$	0
$F_7(x)=\sum_{i=1}^{n} ix_i^4+rand(0,1)$	$[-1.28,1.28]^D$	0

Table 4: Description of multi modal benchmark functions.

Function	Solution space (S)	$f_{\min}$
$F_8(x)=\sum_{i=1}^{n}-x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	$[-500,500]^D$	−418.98
$F_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos\left(2\pi x_i\right)+10\right]$	$[-5.15,5.12]^D$	0
$F_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right)+20+e$	$[-32,32]^D$	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

1 **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
$F_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$	$[-5,5]^D$	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
$F_{19}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right)$	$[0,1]^D$	-3.86
$F_{20}(x)=-\sum_{i=1}^{4}c_i\exp\left(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right)$	$[0,1]^D$	-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

4.2 The performance of TLCO

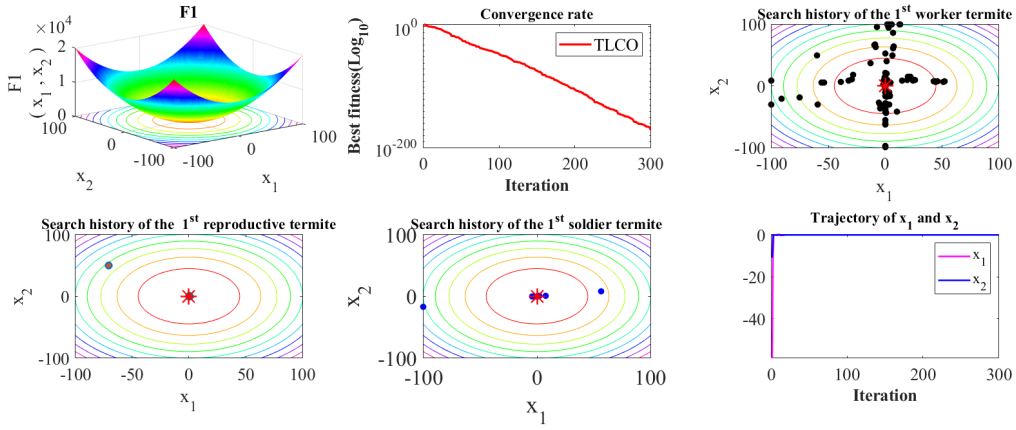
The balance between exploration and exploitation is the primary factor for the successful algorithm. TLCO clearly assign these two skills to termite soldiers and termite workers in the colony, respectively. In addition, reproductive termites will occur when the termite worker have difficulties to find a new food source to establish a new colony. These are the regulation of TLCO to solve optimization problems. Exploration in TLCO is shown in **Fig. 9** (the third column at the first row). It can be noticed that the movement strategy of termite workers covers a wide space during the early iterations. This trend is more evident in the case of functions characterized by multimodality (F7, F17, F19), when the density of the termite workers is almost spread across the whole search space. 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 the control of the step length *S*. A a consequence, the positions of termite workers become monotonous and gradually tend to stabilize to the global optimum in the later iterations. With a flexible movement of termite worker collaborate with the update the information of the best solution (X_{best}) exploited at the previous iteration. Thus, the current best solution (X_{best}) will always orient the expansion of the search space of each termite worker at the next iterations.

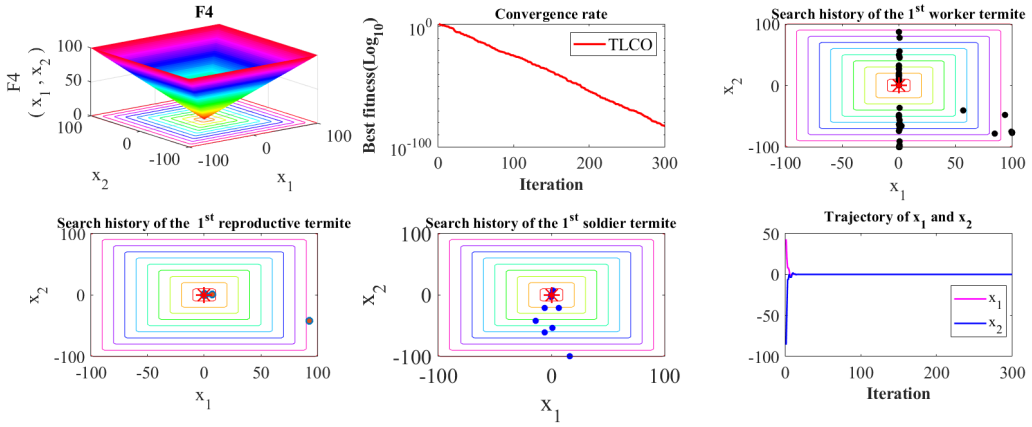
The ability of exploitation in the TLCO is guaranteed by the termite soldiers as shown in the second column at the second row in **Fig. 9**. Note that the processes of exploitation and exploration in TLCO are secured to make a parallel structure at each iteration. Based on the information storage capacity 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, F7, F9, F10 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 F14, F17, F19, F23 in which the best solution is found with much effort. To put it another way, 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.

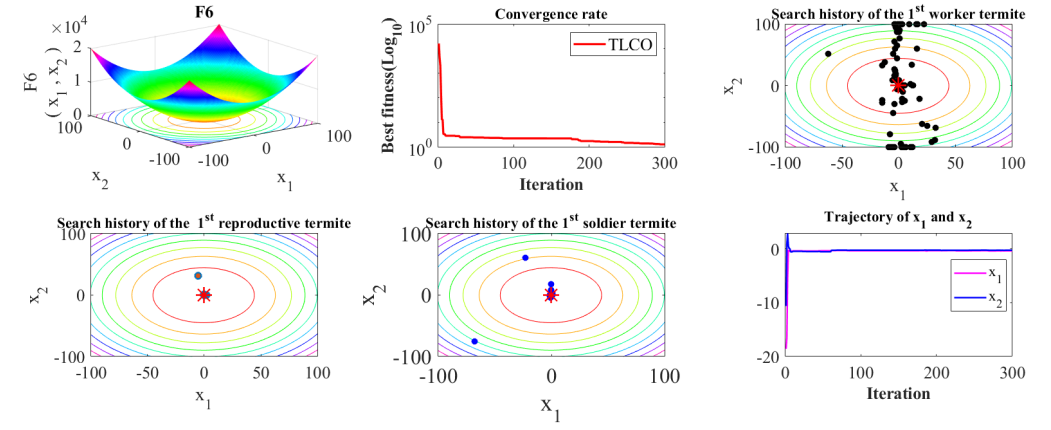
The ability of finding the new colony of reproductive termites is shown in the first column at the second row in Fig. 8. These termites appear to increase the opportunity of exploring new food sources and their effectiveness aren't appreciated in comparison with the worker and termite soldiers. Moreover, they can be ignored in the TLCO by the user through the value of the parameter λ

The trajectory of the first two variables (x_1, x_2) of the best solution (shown in the third column at the second row in **Fig. 9**) is one of the most important metrics used to evaluate the skill of exploration and exploitation in TLCO. All trajectory curves show frequent large fluctuations in the early iterations and reach the stabilization during the later iterations. Evidently, the large fluctuations in the former iterations perform the explorative search for the global space, and the small fluctuations during the latter iterations perform the exploitative search for the local space. TLCO achieves an early convergence rate and acceptable accuracy level in almost all functions. Especially, in functions F1, F4, F9, F19, TLCO just need a few first iterations to find the best solution with a high level of accuracy.

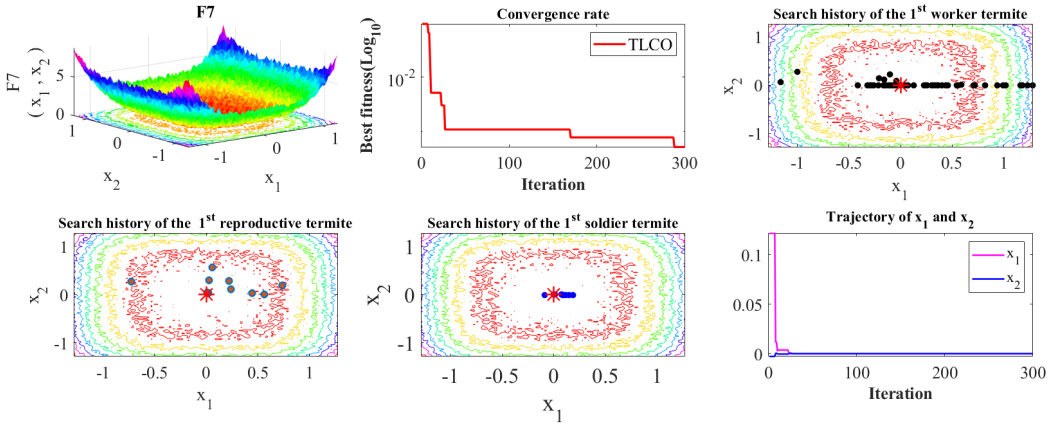
The convergence curve (shown in the second column at the first row in **Fig. 9**) is used to evaluate the convergence performance of the TLCO. The convergence curves of F1, F4, F9, F10, F19 functions are very smooth and dropped rapidly, demonstrating that the skill of exploitation is more biased than 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.



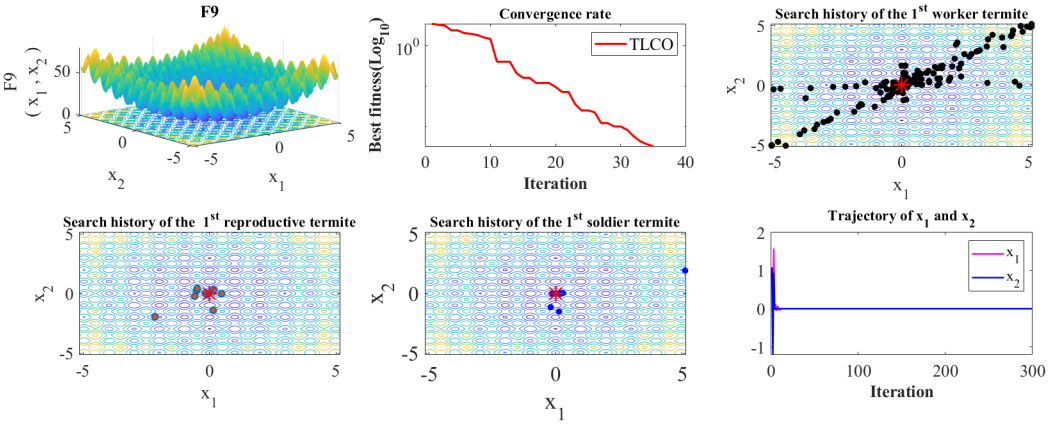



F6
Convergence rate
TLCO
Best fitness(Log10)
Iteration
Search history of the 1st worker termite
Search history of the 1st reproductive termite
Search history of the 1st soldier termite
Trajectory of x1 and x2
x1
x2

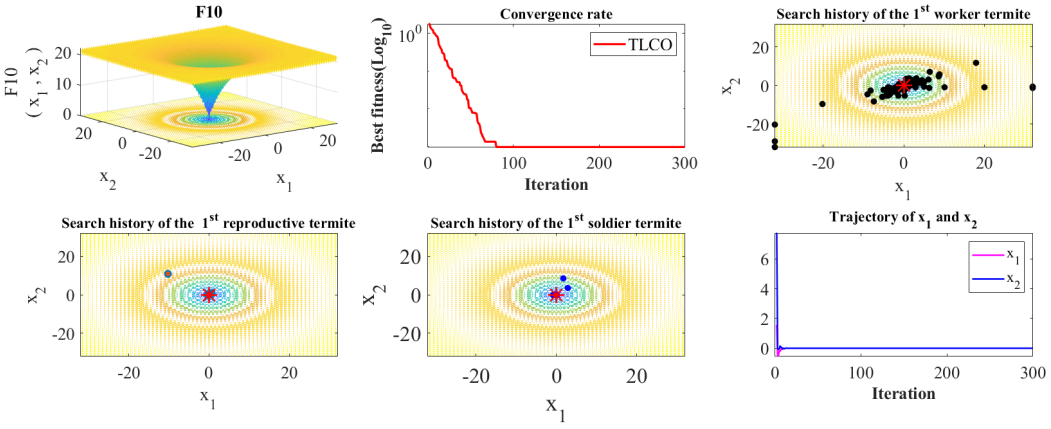
1


F7
Convergence rate
TLCO
Best fitness(Log10)
Iteration
Search history of the 1st worker termite
Search history of the 1st reproductive termite
Search history of the 1st soldier termite
Trajectory of x1 and x2
x1
x2

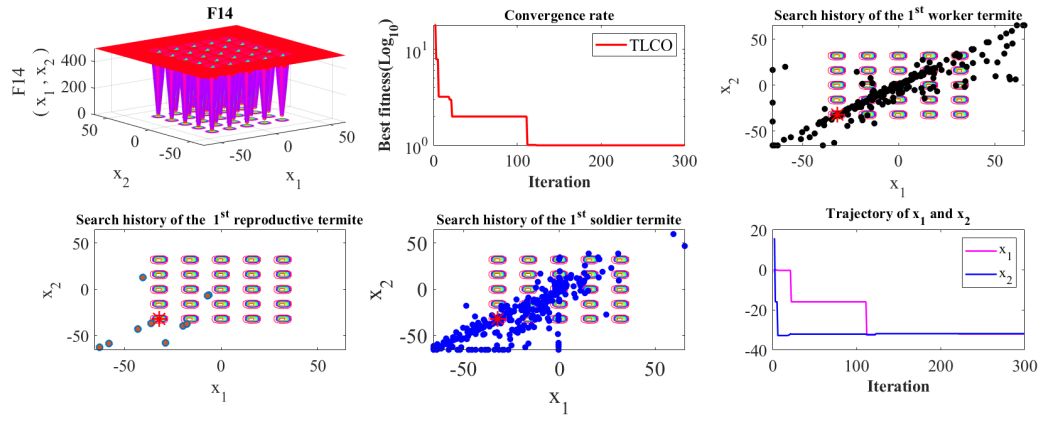
2


F9
Convergence rate
TLCO
Best fitness(Log10)
Iteration
Search history of the 1st worker termite
Search history of the 1st reproductive termite
Search history of the 1st soldier termite
Trajectory of x1 and x2
x1
x2

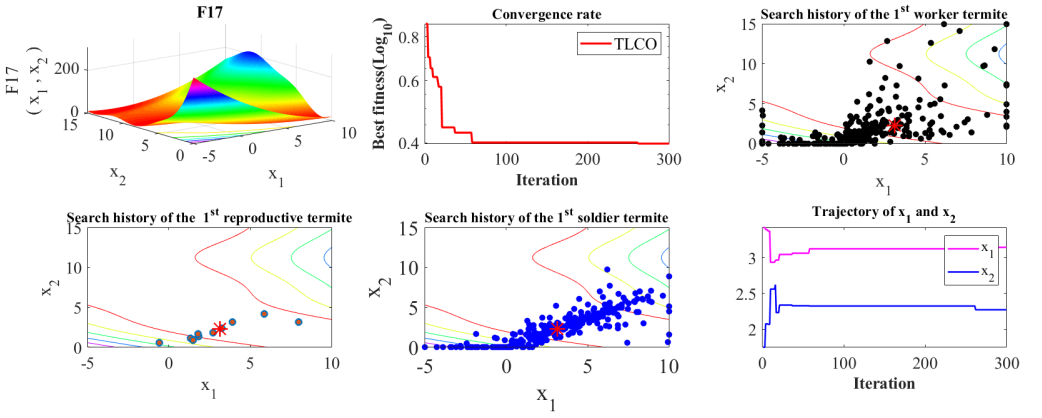
3


F10
Convergence rate
TLCO
Best fitness(Log10)
Iteration
Search history of the 1st worker termite
Search history of the 1st reproductive termite
Search history of the 1st soldier termite
Trajectory of x1 and x2
x1
x2

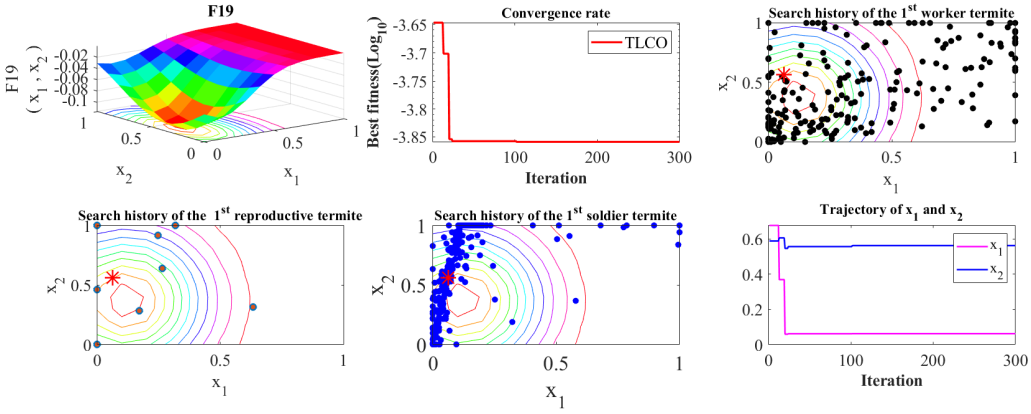
4


F14
F14 (x1 , x2)
x2
x1
Convergence rate
TLCO
Best fitness(Log10)
Iteration
Search history of the 1st worker termite
Search history of the 1st reproductive termite
Search history of the 1st soldier termite
Trajectory of x1 and x2
x1
x2

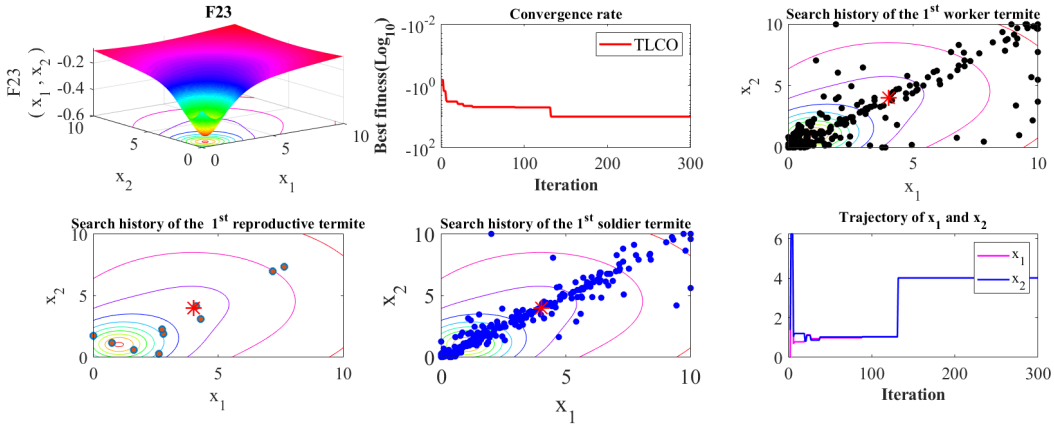
1


F17
F17 (x1 , x2)
x2
x1
Convergence rate
TLCO
Best fitness(Log10)
Iteration
Search history of the 1st worker termite
Search history of the 1st reproductive termite
Search history of the 1st soldier termite
Trajectory of x1 and x2
x1
x2

2


F19
F19 (x1 , x2)
x2
x1
Convergence rate
TLCO
Best fitness(Log10)
Iteration
Search history of the 1st worker termite
Search history of the 1st reproductive termite
Search history of the 1st soldier termite
Trajectory of x1 and x2
x1
x2

3


F23
F23 (x1 , x2)
x2
x1
Convergence rate
TLCO
Best fitness(Log10)
Iteration
Search history of the 1st worker termite
Search history of the 1st reproductive termite
Search history of the 1st soldier termite
Trajectory of x1 and x2
x1
x2

4

Fig. 9: Quantitative results of TLCO for three groups of typical functions.

4.3 The comparison between TLCO with other algorithms

An efficient algorithm must demonstrate its ability to find the best solution in search spaces of different dimension, especially, with large-scale dimension 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 $D = 30$, $D = 50$ and $D = 100$. In the case of functions (F14-F23), the dimensions are fixed as shown in **Table 5**.

Fig. 10. shows the results obtained by TLCO compared to 8 well-known algorithms: PSO [38], GA [12], GSA [23], (FA) [53], DE [18], (ASO) [31], (GWO) [59], and (CS) [50]. For a fair comparison, all algorithms mentioned above have been tested with the same initial conditions including the dimension of the search space $D = 50$, the number of particles $N = 30$ and the total number of iterations 1000. Then, the values of four items including the best values, the worst value, the mean values, the standard deviation values obtained for each algorithm are presented.. **Table 6**, **Table 7** and

Table *8*, show the results obtained with 30 dimensions, 50 dimensions and 100 dimensions, respectively.

The necessary parameters of all considered algorithms are set as follows:

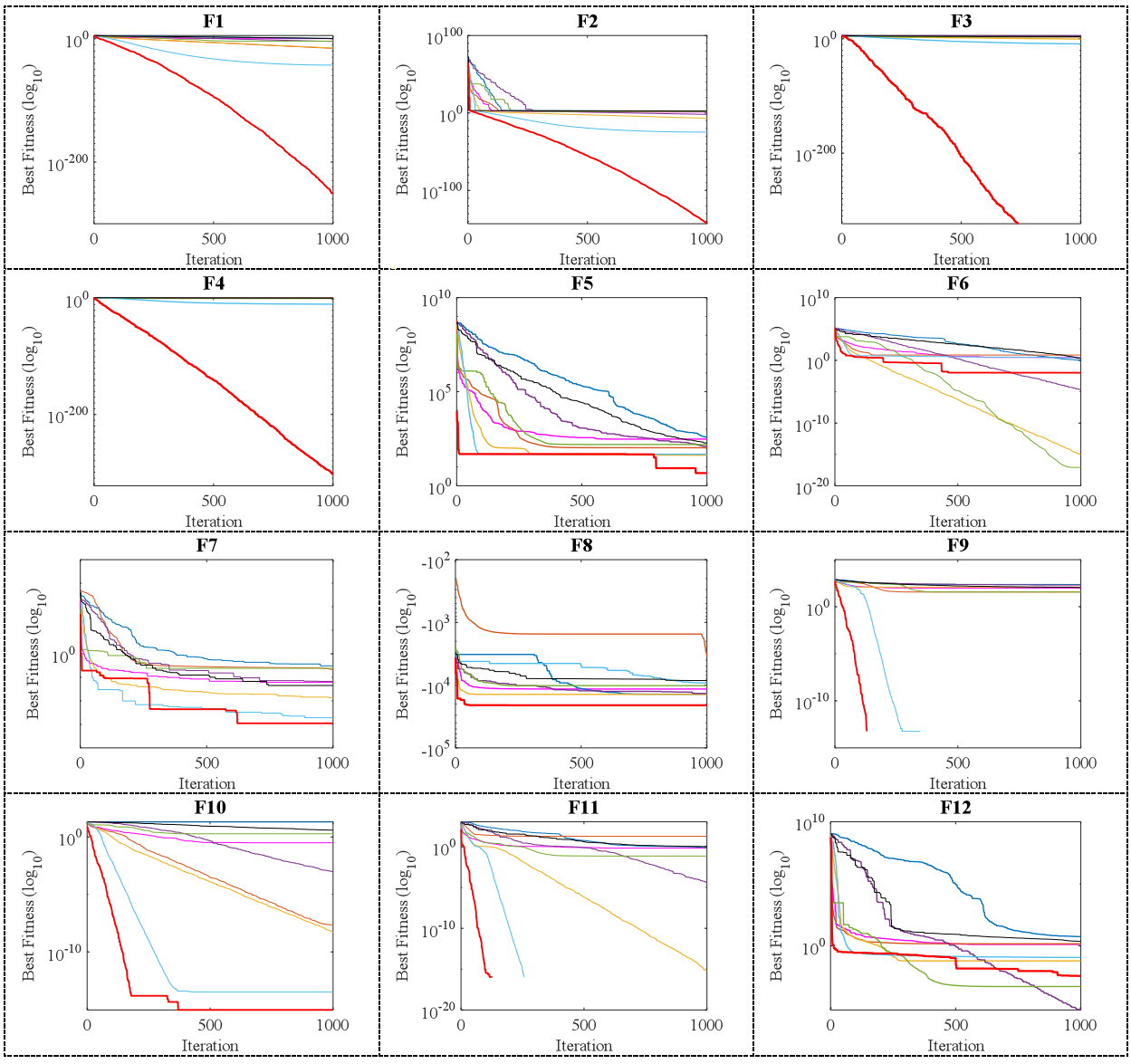
- In TLCO, the parameter used to specify the time when reproductive termites emerge is $\mu = 0.1$
- In PSO, the values of cognitive and social parameters were $c_1 = 2$ and $c_2 = 2$, respectively, and the inertia weight w increased from 0.4 to 0.9.
- In GA, we used the parameters of crossover were $\mu = 0.8$, and mutation $m = 0.4$
- In GSA, we used a gravitational constant $G = 100$ and a decreasing coefficient $\beta = 20$.
- In FA, we used a light absorption coefficient $\gamma = 0.1$ and an attraction coefficient base value $\beta_0 = 0.2$,
- In DE, we used a mutation factor $CR = 0.9$ and a crossover probability $F = 0.5$
- In ASO, we used a depth weigh = 50 and a multiplier weight = 0.2.
- For GWO and CS we used default parameters of the original versions [59], [50]

Although the results shown in **Fig. 9** reveal that TLCO has good results, to get a more exhaustive view, **Fig. 10** shows the convergence curve of each algorithm in the same figure. The convergence curve is one of the most significant evaluations that effectively quantify the exploration and exploitation abilities of each algorithm.

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 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 through the control of step length *S*. As a result, TLCO shows better performances in convergence rate compared to other algorithms in the case of functions F1, F2, F3, F4, F5, F7, F9, F10, F11 because of the long movements performed in the first few iterations. However, TLCO fails to achieve the best convergence performance the case of functions F6, F8, F12, F13. However, TLCO still performs better than PSO, GA, GSA, GWO, CS for F3, PSO, GSA, DE, ASO, GWO, CS for F6, PSO, GA, GSA, DE, ASO, GWO, CS for F12 and 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 find and exploit the best solution. However, the convergence rate of TLCO reaches stability faster in almost functions compared to other algorithms with the exception of functions 21, 22 and 23. Especially as regards the functions F14, F15, F16, F17, F18, F19 and F20, TLCO only requires a few iterations to achieve the stability. This proves that the TLCO's ability to escape local optima is better than other algorithms that need more iterations to achieve the necessary stability.



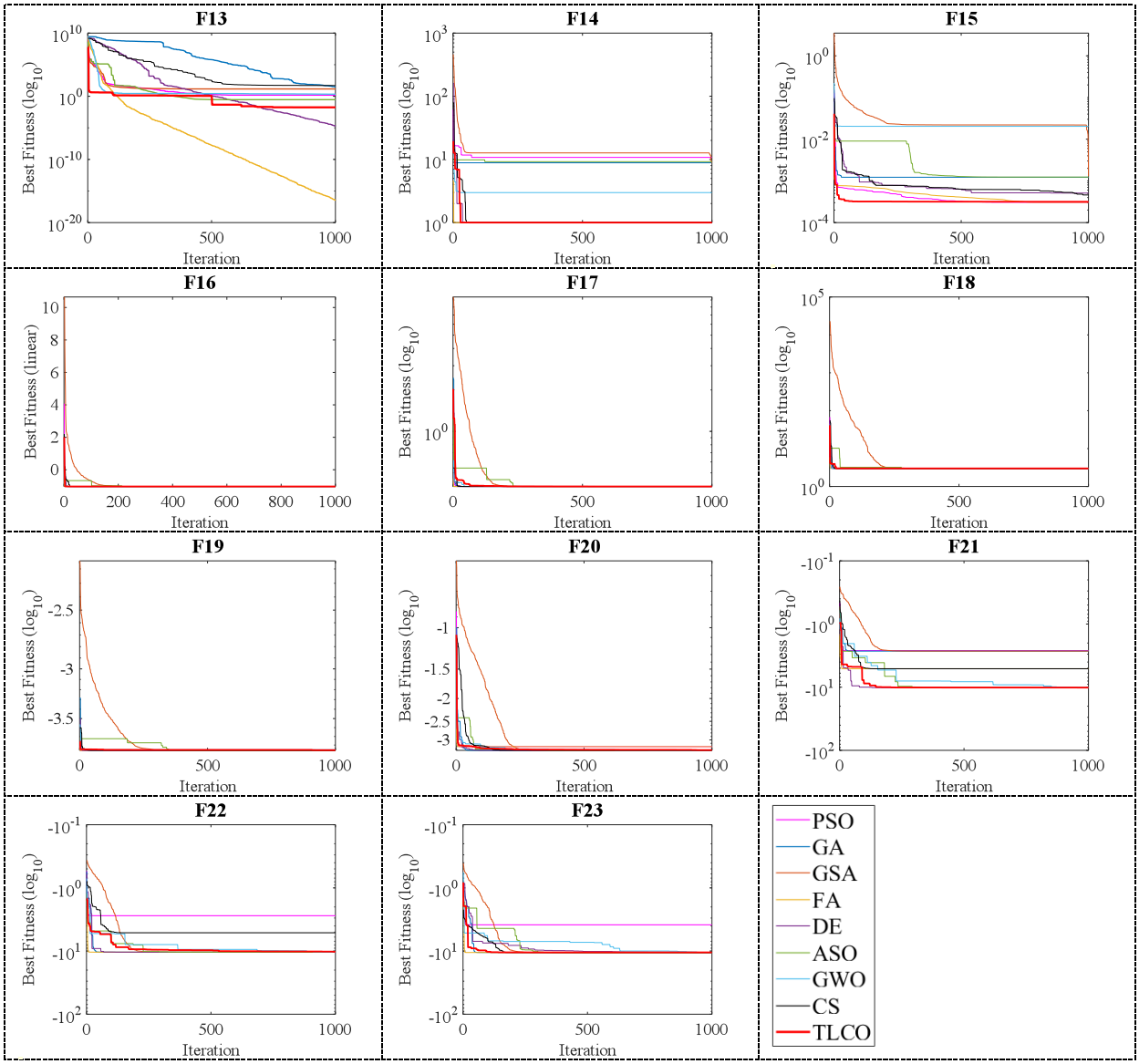


Fig. 10: Convergence trends of the 23 benchmark functions with different algorithms

Table 6 summarizes the results obtained for the different tested algorithms with dimension D = 30. One can notice that TLCO performs better than other algorithms in the case of functions F1, F2, F3, F4, F7, F9, F10 and F11. Even the worst value obtained with TLCO is better than the best value obtained with other algorithms. As regards function F5 for which the best performance is obtained with FA, the performance of TLCO is still better than the one obtained with other algorithms in terms of the worst value, the mean value and the standard deviation.

The results obtained with a search space of dimension D = 50 are reported in Table 7 and show a similar trend to the one observed in a search space of dimension D = 30. It can be observed that TLCO reaches a robust enhancement as regard function F12. In the case of dimension D = 30, TLCO fails to achieve the best performance for this function, however, in a higher dimension (when D = 50), TLCO improves its rankings: it ranks third in term of the best value, behind FA, DE and ASO algorithms.

For even higher dimension of the search space, when D = 100, a common characteristic can be observed when the total number of iterations is set to 1000; in that case, the number of iterations is not large enough for algorithms to find the best global value with the exception of TLCO 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. The best value using TLCO is ranked first for the functions F1, F2, F4, F7, F8, F9, F10, F11, F12 and F13. Especially, TLCO's ranking continues to improve for the functions F12 and F13 to reach the first ranking. It appears clearly that TLCO is efficient in high-dimension of case studies. Usually, the algorithms will have difficulty in high-dimension search space because position updating does not guarantee a flexible movement.

As regards the functions with fixed variables (F14-F23) whose main characteristic is that the number of variables is limited with low-dimension, it can be seen that all algorithms can find the best value within 1000 iterations. Not a single algorithm really performs better than the others in this case and TLCO still finds the best value along with other algorithms. The only difference is that the convergence rate of the TLCO is better in some particular functions as shown in **Fig. 10**.

Table 6: Comparison results of the first 13 benchmark functions (F1-F13) with dimension $D = 30$, where the best performance for each algorithm is highlighted in red

Function		Different Algorithms								
		TLCO	PSO	GA	GSA	FA	DE	ASO	GWO	CS
F1	Best	0.000E+00	3.215E-05	2.861E-06	5.017E-17	1.173E-16	7.525E-13	2.478E-22	7.186E-62	2.378E-04
	Worst	0.000E+00	4.391E-03	1.691E-04	2.004E-16	2.036E-16	1.283E-11	3.027E-18	4.633E-58	1.060E-03
	Mean	0.000E+00	6.460E-04	3.947E-05	1.181E-16	1.749E-16	3.051E-12	1.083E-19	6.592E-59	4.401E-04
	S. Deviation	0.000E+00	8.505E-04	3.191E-05	5.379E-17	2.326E-17	2.320E-12	5.513E-19	1.428E-58	2.550E-04
F2	Best	0.000E+00	7.552E-03	1.973E-03	4.039E-08	4.744E-08	3.390E-07	1.092E-10	2.980E-34	9.097E+01
	Worst	2.516E-294	2.624E+02	1.600E-01	1.113E-07	6.102E-08	1.182E-06	6.002E+01	4.868E-33	1.682E+02
	Mean	2.516E-295	1.744E+01	3.310E-02	5.808E-08	5.628E-08	6.449E-07	1.562E+01	1.410E-33	1.241E+02
	S. Deviation	0.000E+00	6.612E+01	3.697E-02	2.126E-08	4.401E-09	2.195E-07	1.643E+01	1.474E-33	2.734E+01
F3	Best	0.000E+00	1.230E+02	4.541E+02	1.915E+02	9.354E-12	1.932E+04	1.095E+02	4.876E-20	4.631E+02
	Worst	0.000E+00	8.147E+02	5.392E+03	6.384E+02	8.217E-09	3.195E+04	7.519E+02	4.800E-14	7.539E+02
	Mean	0.000E+00	4.076E+02	1.905E+03	4.563E+02	1.712E-09	2.503E+04	2.862E+02	7.784E-15	5.933E+02
	S. Deviation	0.000E+00	1.532E+02	1.190E+03	1.453E+02	2.006E-09	3.474E+03	1.728E+02	1.584E-14	9.156E+01
F4	Best	1.164E-306	2.865E+00	3.000E+01	1.336E-08	5.540E-09	1.039E+00	2.611E-03	2.174E-15	3.699E+00
	Worst	2.236E-286	6.180E+00	7.803E+01	5.543E+00	6.605E+00	3.667E+00	7.677E-01	3.204E-14	1.262E+01
	Mean	3.387E-287	4.239E+00	6.043E+01	1.046E+00	1.830E+00	2.057E+00	3.081E-01	1.238E-14	5.805E+00
	S. Deviation	0.000E+00	7.716E-01	1.025E+01	1.770E+00	1.978E+00	4.932E-01	2.410E-01	9.993E-15	2.528E+00
F5	Best	2.589E+01	1.808E+01	1.694E+01	2.588E+01	1.393E+01	2.614E+01	2.446E+01	2.598E+01	2.365E+01
	Worst	2.705E+01	2.164E+02	4.355E+02	1.247E+02	8.135E+01	1.097E+02	1.852E+02	2.720E+01	1.165E+02
	Mean	2.657E+01	7.740E+01	8.604E+01	5.115E+01	2.744E+01	4.696E+01	5.526E+01	2.685E+01	4.176E+01
	S. Deviation	4.221E-01	5.493E+01	8.675E+01	4.113E+01	1.848E+01	2.523E+01	4.425E+01	4.954E-01	3.230E+01
F6	Best	4.072E-02	3.753E-05	3.651E-06	0.000E+00	1.404E-16	6.054E-13	2.722E-22	2.483E-01	2.139E-04
	Worst	3.362E-01	1.429E-03	4.037E-04	0.000E+00	2.187E-16	1.171E-11	6.251E-20	1.002E+00	1.434E-03
	Mean	1.489E-01	4.705E-04	5.228E-05	0.000E+00	1.780E-16	3.292E-12	6.183E-21	5.184E-01	6.312E-04
	S. Deviation	1.098E-01	4.422E-04	7.751E-05	0.000E+00	2.301E-17	2.293E-12	1.231E-20	2.434E-01	3.353E-04
F7	Best	1.813E-05	6.218E-03	2.051E-02	5.691E-02	7.652E-04	1.594E-02	3.597E-02	5.903E-04	1.762E-02
	Worst	3.338E-04	3.705E-02	9.104E-02	2.563E-01	5.384E-03	4.174E-02	1.416E-01	1.371E-03	3.942E-02
	Mean	1.661E-04	1.970E-02	4.914E-02	1.414E-01	2.351E-03	2.635E-02	8.634E-02	1.026E-03	2.451E-02
	S. Deviation	1.117E-04	8.768E-03	1.755E-02	5.724E-02	1.122E-03	6.223E-03	2.473E-02	2.637E-04	6.951E-03
F8	Best	-1.233E+04	-8.085E+03	-9.883E+03	-1.774E+03	-1.042E+04	-1.257E+04	-8.463E+03	-6.379E+03	-7.018E+03
	Worst	-8.874E+03	-4.594E+03	-7.124E+03	-1.134E+03	-7.829E+03	-1.221E+04	-4.238E+03	-5.525E+03	-5.771E+03
	Mean	-1.044E+04	-6.312E+03	-8.283E+03	-1.342E+03	-9.107E+03	-1.245E+04	-6.512E+03	-6.020E+03	-6.443E+03
	S. Deviation	1.061E+03	8.596E+02	6.341E+02	1.821E+02	6.211E+02	1.260E+02	1.105E+03	3.312E+02	4.293E+02
F9	Best	0.000E+00	1.995E+01	7.263E+01	1.890E+01	7.462E+01	4.706E+01	1.592E+01	0.000E+00	3.569E+01
	Worst	0.000E+00	8.265E+01	2.627E+02	4.477E+01	2.189E+02	7.102E+01	4.378E+01	5.684E-14	6.262E+01
	Mean	0.000E+00	4.019E+01	1.648E+02	3.283E+01	1.446E+02	6.066E+01	2.511E+01	1.137E-14	4.713E+01

	S. Deviation	0.000E+00	1.335E+01	4.671E+01	8.311E+00	3.508E+01	6.063E+00	6.706E+00	2.397E-14	8.572E+00
F10	Best	8.882E-16	1.680E-03	2.678E-03	6.770E-09	2.635E-09	2.650E-07	4.953E-12	1.155E-14	1.531E-01
	Worst	8.882E-16	1.043E+00	1.996E+01	9.404E-09	3.489E-09	1.125E-06	1.340E+00	2.220E-14	2.880E+00
	Mean	8.882E-16	6.574E-02	1.495E+01	8.032E-09	3.150E-09	4.968E-07	4.468E-02	1.759E-14	1.363E+00
	S. Deviation	0.000E+00	1.946E-01	8.394E+00	8.795E-10	2.392E-10	1.974E-07	2.447E-01	4.119E-15	8.256E-01
F11	Best	0.000E+00	1.637E-04	2.522E-06	3.420E+00	0.000E+00	3.796E-12	0.000E+00	0.000E+00	4.235E-03
	Worst	0.000E+00	9.909E-02	3.923E-02	1.285E+01	1.477E-02	2.566E-10	8.532E-02	1.393E-02	2.026E-01
	Mean	0.000E+00	1.681E-02	1.033E-02	8.656E+00	3.080E-03	5.186E-11	9.737E-03	2.738E-03	6.491E-02
	S. Deviation	0.000E+00	2.253E-02	1.063E-02	3.341E+00	5.172E-03	6.806E-11	1.836E-02	5.774E-03	6.877E-02
F12	Best	1.176E-03	2.221E-07	2.273E-05	4.464E-19	4.298E-19	8.692E-14	1.097E-24	2.043E-02	9.128E-03
	Worst	2.174E-02	2.075E-01	2.385E+00	2.073E-01	4.147E-01	8.620E-13	2.073E-01	8.563E-02	1.832E+00
	Mean	6.952E-03	3.729E-02	4.121E-01	5.976E-02	6.220E-02	2.757E-13	6.911E-03	4.451E-02	1.006E+00
	S. Deviation	6.377E-03	6.211E-02	5.652E-01	6.905E-02	1.085E-01	1.578E-13	3.785E-02	1.992E-02	6.624E-01
F13	Best	4.535E-02	1.140E-05	7.026E-06	6.383E-18	5.663E-18	3.876E-13	1.100E-23	1.141E-01	1.695E-02
	Worst	3.431E-01	1.218E-02	3.598E+00	6.664E-01	1.099E-02	6.130E-12	9.737E-02	7.351E-01	2.278E+00
	Mean	1.243E-01	3.458E-03	3.585E-01	6.664E-02	5.494E-04	1.745E-12	7.274E-03	3.897E-01	9.159E-01
	S. Deviation	9.182E-02	5.018E-03	8.760E-01	2.107E-01	2.457E-03	1.331E-12	1.918E-02	1.840E-01	8.425E-01

Table 7: Comparison results of the first 13 benchmark functions (F1-F13) with dimension $D = 50$, where the best performance for each algorithm is highlighted in red

Function		Different Algorithms								
		TLCO	PSO	GA	GSA	FA	DE	ASO	GWO	CS
F1	Best	0.000E+00	3.321E-01	1.116E-01	3.222E-16	7.061E-16	1.172E-05	1.189E-17	4.466E-45	1.035E+00
	Worst	0.000E+00	5.644E+00	5.551E+00	1.671E-15	1.152E-15	3.837E-05	2.356E-01	2.385E-42	7.607E+00
	Mean	0.000E+00	1.502E+00	1.109E+00	6.121E-16	8.856E-16	2.009E-05	1.263E-02	1.837E-43	2.971E+00
	S. Deviation	0.000E+00	2.616E+02	5.213E+01	5.872E-01	1.020E-16	2.685E-03	5.897E+01	3.409E-25	4.120E+01
F2	Best	0.000E+00	3.205E+00	3.587E+00	1.475E-07	1.396E-07	5.899E-03	5.725E+01	1.377E-25	2.324E+02
	Worst	8.768E-297	8.775E+02	1.401E+02	2.628E+00	1.714E-07	1.602E-02	2.832E+02	1.508E-24	3.646E+02
	Mean	4.386E-298	1.133E+02	4.311E+01	1.750E-01	1.562E-07	9.739E-03	1.477E+02	4.516E-25	3.087E+02
	S. Deviation	0.000E+00	1.513E+03	6.399E+03	4.640E+02	9.684E-09	1.140E+04	5.103E+02	6.506E-06	1.031E+03
F3	Best	0.000E+00	2.114E+03	1.158E+04	1.263E+03	8.733E-02	6.404E+04	1.435E+03	7.760E-11	3.576E+03
	Worst	0.000E+00	8.328E+03	3.343E+04	3.300E+03	5.158E+00	1.074E+05	3.338E+03	2.443E-05	7.022E+03
	Mean	0.000E+00	4.273E+03	2.436E+04	1.883E+03	9.552E-01	9.251E+04	2.279E+03	2.634E-06	5.047E+03
	S. Deviation	0.000E+00	1.171E+00	5.051E+00	1.918E+00	1.097E+00	2.286E+00	2.206E+00	1.943E-09	2.492E+00
F4	Best	2.838E-302	7.467E+00	6.188E+01	5.404E+00	1.734E+01	1.771E+01	3.608E+00	1.467E-10	9.041E+00
	Worst	4.098E-284	1.262E+01	8.352E+01	1.345E+01	5.001E+01	2.579E+01	1.153E+01	7.431E-09	1.799E+01
	Mean	2.056E-285	9.908E+00	7.657E+01	8.616E+00	3.139E+01	2.102E+01	7.272E+00	1.934E-09	1.336E+01
	S. Deviation	0.000E+00	1.254E+02	6.949E+02	5.414E+01	7.401E+00	8.003E+01	9.015E+01	8.012E-01	1.344E+02

F5	Best	4.615E+01	1.756E+02	2.512E+02	4.675E+01	2.192E+01	4.798E+01	4.652E+01	4.584E+01	5.891E+01
	Worst	4.838E+01	6.232E+02	3.141E+03	2.154E+02	1.093E+02	3.248E+02	4.230E+02	4.862E+01	6.293E+02
	Mean	4.714E+01	3.797E+02	7.199E+02	1.078E+02	5.420E+01	1.681E+02	1.100E+02	4.685E+01	2.277E+02
	S. Deviation	8.344E-01	1.254E+02	6.949E+02	5.414E+01	2.768E+01	8.003E+01	9.015E+01	8.012E-01	1.344E+02
F6	Best	5.086E-01	2.110E-01	1.503E-01	3.000E+00	6.624E-16	7.458E-06	4.275E-18	1.252E+00	9.741E-01
	Worst	2.225E+00	7.052E+00	4.943E+00	1.920E+02	1.008E-15	4.299E-05	2.744E-02	3.763E+00	6.168E+00
	Mean	1.153E+00	2.033E+00	9.016E-01	3.155E+01	8.251E-16	2.377E-05	2.240E-03	2.496E+00	2.354E+00
	S. Deviation	4.787E-01	1.762E+00	1.187E+00	4.289E+01	1.087E-16	1.106E-05	6.855E-03	7.533E-01	1.265E+00
F7	Best	1.952E-05	4.265E-02	1.019E-01	9.584E-02	5.430E-03	5.220E-02	1.226E-01	4.518E-04	3.277E-02
	Worst	3.297E-04	8.665E-02	3.284E-01	5.029E-01	2.236E-02	1.133E-01	2.455E-01	4.009E-03	1.248E-01
	Mean	1.369E-04	6.287E-02	2.089E-01	2.735E-01	1.110E-02	7.969E-02	1.794E-01	1.414E-03	8.290E-02
	S. Deviation	1.093E-04	1.376E-02	5.613E-02	1.067E-01	4.187E-03	1.419E-02	3.718E-02	9.303E-04	2.533E-02
F8	Best	-2.091E+04	-1.228E+04	-1.452E+04	-2.266E+03	-1.579E+04	-1.525E+04	-1.328E+04	-1.092E+04	-9.681E+03
	Worst	-1.302E+04	-8.345E+03	-1.208E+04	-1.027E+03	-1.214E+04	-1.296E+04	-8.049E+03	-7.861E+03	-7.593E+03
	Mean	-1.730E+04	-1.010E+04	-1.360E+04	-1.599E+03	-1.415E+04	-1.399E+04	-1.055E+04	-8.981E+03	-8.494E+03
	S. Deviation	2.733E+03	1.201E+03	5.705E+02	2.882E+02	9.367E+02	5.778E+02	1.317E+03	8.705E+02	5.273E+02
F9	Best	0.000E+00	3.831E+01	2.043E+02	2.985E+01	6.766E+01	1.586E+02	3.184E+01	0.000E+00	7.368E+01
	Worst	0.000E+00	1.094E+02	4.698E+02	7.860E+01	2.328E+02	2.165E+02	6.368E+01	1.027E+01	1.376E+02
	Mean	0.000E+00	6.410E+01	3.239E+02	5.199E+01	1.303E+02	1.963E+02	4.826E+01	1.591E+00	1.027E+02
	S. Deviation	0.000E+00	1.819E+01	7.062E+01	1.339E+01	4.040E+01	1.401E+01	6.330E+00	3.096E+00	1.725E+01
F10	Best	8.882E-16	3.337E-01	2.119E+00	1.090E-08	4.835E-09	6.920E-04	2.898E-01	2.576E-14	2.671E+00
	Worst	8.882E-16	2.116E+00	1.996E+01	1.270E+00	5.789E-09	1.175E-03	2.013E+00	3.997E-14	5.646E+00
	Mean	8.882E-16	1.432E+00	1.869E+01	2.380E-01	5.308E-09	9.375E-04	1.478E+00	3.269E-14	3.964E+00
	S. Deviation	0.000E+00	5.499E-01	3.948E+00	4.595E-01	2.572E-10	1.332E-04	4.615E-01	3.356E-15	8.352E-01
F11	Best	0.000E+00	1.597E-01	1.265E-01	2.145E+01	1.110E-16	1.190E-05	1.867E-04	0.000E+00	9.920E-01
	Worst	0.000E+00	9.962E-01	1.043E+00	5.314E+01	7.396E-03	6.503E-04	1.088E+00	2.673E-02	1.099E+00
	Mean	0.000E+00	6.324E-01	5.926E-01	3.373E+01	3.698E-04	6.473E-05	4.038E-01	2.404E-03	1.048E+00
	S. Deviation	0.000E+00	2.117E-01	2.915E-01	8.189E+00	1.654E-03	1.385E-04	3.707E-01	6.604E-03	2.677E-02
F12	Best	7.470E-03	1.297E-02	1.034E+00	6.715E-01	9.564E-19	3.909E-06	7.512E-05	4.952E-02	2.089E+00
	Worst	6.901E-02	1.231E+00	1.767E+01	3.144E+00	1.811E+00	3.128E-05	6.501E-01	1.725E-01	5.999E+00
	Mean	2.586E-02	4.695E-01	5.950E+00	1.686E+00	4.029E-01	1.034E-05	1.305E-01	8.878E-02	3.756E+00
	S. Deviation	1.635E-02	3.859E-01	4.254E+00	7.847E-01	5.441E-01	6.659E-06	1.577E-01	3.142E-02	1.080E+00
F13	Best	2.497E-01	9.478E-01	6.154E+00	1.251E+00	2.862E-17	2.968E-05	1.660E-18	1.026E+00	2.044E+01
	Worst	5.793E-01	3.425E+00	4.175E+01	3.932E+01	1.099E-02	1.976E-04	1.038E+00	2.754E+00	5.317E+01
	Mean	3.920E-01	1.931E+00	2.008E+01	1.816E+01	3.846E-03	6.232E-05	1.881E-01	1.958E+00	3.566E+01
	S. Deviation	1.021E-01	6.578E-01	9.549E+00	9.889E+00	5.377E-03	3.884E-05	2.955E-01	4.549E-01	9.471E+00

Table 8: Comparison results of the first 13 benchmark functions (F1-F13) with dimension $D = 100$, where the best performance for each algorithm is highlighted in red

Function		Different Algorithms								
		TLCO	PSO	GA	GSA	FA	DE	ASO	GWO	CS
F1	Best	0.000E+00	1.316E+02	1.730E+03	2.192E+02	9.762E-15	9.175E+00	1.633E+02	1.571E-30	3.441E+02
	Worst	0.000E+00	5.016E+02	5.369E+03	1.364E+03	1.843E-14	1.612E+01	1.027E+03	5.983E-29	8.367E+02
	Mean	0.000E+00	2.610E+02	3.057E+03	6.298E+02	1.372E-14	1.237E+01	4.229E+02	1.779E-29	5.945E+02
	S. Deviation	0.000E+00	8.514E+01	7.801E+02	3.099E+02	2.476E-15	2.008E+00	1.827E+02	1.445E-29	1.430E+02
F2	Best	0.000E+00	2.230E+02	1.081E+03	3.645E+00	6.460E-07	6.851E+01	6.881E+02	2.345E-17	7.975E+02
	Worst	1.234E-295	1.485E+03	2.752E+03	9.345E+00	3.057E-02	1.584E+02	1.058E+03	1.311E-16	1.025E+03
	Mean	6.465E-297	5.111E+02	1.631E+03	5.880E+00	1.574E-03	1.048E+02	9.067E+02	6.963E-17	8.884E+02
	S. Deviation	0.000E+00	3.350E+02	3.814E+02	1.871E+00	6.829E-03	2.338E+01	1.090E+02	3.278E-17	6.027E+01
F3	Best	0.000E+00	1.759E+04	1.006E+05	5.572E+03	2.566E+03	3.214E+05	1.153E+04	7.289E-03	2.581E+04
	Worst	2.382E-255	4.868E+04	2.241E+05	1.116E+04	6.668E+03	4.432E+05	2.900E+04	1.577E+01	5.287E+04
	Mean	1.191E-256	2.967E+04	1.480E+05	8.092E+03	4.469E+03	3.982E+05	1.792E+04	2.409E+00	3.948E+04
	S. Deviation	0.000E+00	8.044E+03	3.199E+04	1.454E+03	1.082E+03	2.800E+04	4.652E+03	4.119E+00	7.443E+03
F4	Best	4.293E-300	1.899E+01	7.977E+01	1.200E+01	6.641E+01	7.944E+01	1.372E+01	2.659E-05	1.683E+01
	Worst	1.870E-281	2.536E+01	9.263E+01	1.969E+01	9.762E+01	8.653E+01	2.321E+01	2.509E-02	2.737E+01
	Mean	9.632E-283	2.120E+01	8.705E+01	1.589E+01	8.547E+01	8.307E+01	1.967E+01	3.235E-03	2.198E+01
	S. Deviation	0.000E+00	1.686E+00	3.139E+00	1.978E+00	7.125E+00	2.174E+00	2.466E+00	5.764E-03	2.522E+00
F5	Best	9.653E+01	1.148E+04	4.558E+05	3.684E+03	1.022E+02	9.576E+03	8.933E+02	9.602E+01	7.843E+03
	Worst	9.841E+01	3.256E+04	3.363E+06	3.627E+04	3.181E+02	3.143E+04	5.231E+03	9.845E+01	5.217E+04
	Mean	9.754E+01	1.862E+04	1.188E+06	1.201E+04	1.978E+02	1.747E+04	2.243E+03	9.768E+01	2.297E+04
	S. Deviation	6.695E-01	5.400E+03	6.283E+05	7.878E+03	5.174E+01	6.443E+03	9.826E+02	6.465E-01	1.045E+04
F6	Best	3.290E+00	6.780E-05	4.378E-06	7.770E+02	9.691E-15	1.032E-12	8.035E-23	2.501E-01	1.422E-04
	Worst	8.422E+00	2.889E-03	1.454E-04	2.487E+03	1.763E-14	8.023E-12	8.789E-20	1.248E+00	8.980E-04
	Mean	5.981E+00	4.703E-04	4.827E-05	1.425E+03	1.273E-14	2.848E-12	1.008E-20	6.229E-01	5.849E-04
	S. Deviation	1.432E+00	6.305E-04	4.544E-05	5.252E+02	1.644E-15	1.615E-12	2.383E-20	2.853E-01	1.831E-04
F7	Best	1.801E-05	2.553E-01	1.114E+00	9.608E-01	5.184E-02	4.007E-01	3.760E-01	7.718E-04	2.371E-01
	Worst	1.168E-03	5.535E-01	3.010E+00	4.405E+00	1.236E-01	6.798E-01	1.021E+00	5.751E-03	3.812E-01
	Mean	2.203E-04	4.185E-01	1.843E+00	2.436E+00	8.036E-02	5.389E-01	6.678E-01	2.799E-03	3.037E-01
	S. Deviation	2.751E-04	8.604E-02	4.346E-01	9.753E-01	2.051E-02	5.739E-02	1.541E-01	1.243E-03	3.928E-02
F8	Best	-4.165E+04	-2.356E+04	-2.892E+04	-1.383E+03	-2.872E+04	-1.986E+04	-2.214E+04	-1.766E+04	-1.347E+04
	Worst	-2.481E+04	-1.511E+04	-2.467E+04	-3.425E+02	-2.098E+04	-1.771E+04	-1.390E+04	-6.888E+03	-1.078E+04
	Mean	-3.687E+04	-1.935E+04	-2.663E+04	-9.862E+02	-2.589E+04	-1.846E+04	-1.870E+04	-1.587E+04	-1.217E+04
	S. Deviation	5.266E+03	2.307E+03	1.134E+03	2.998E+02	1.769E+03	5.990E+02	2.194E+03	2.278E+03	6.987E+02
F9	Best	0.000E+00	1.168E+02	6.273E+02	7.875E+01	2.438E+02	6.493E+02	8.230E+01	1.137E-13	2.118E+02
	Worst	0.000E+00	2.529E+02	9.607E+02	1.671E+02	4.806E+02	7.199E+02	1.638E+02	8.867E+00	3.518E+02
	Mean	0.000E+00	1.620E+02	7.578E+02	1.393E+02	3.800E+02	6.902E+02	1.207E+02	6.693E-01	2.705E+02

	S. Deviation	0.000E+00	3.349E+01	9.407E+01	2.209E+01	6.158E+01	2.276E+01	2.115E+01	2.051E+00	3.302E+01
F10	Best	8.882E-16	3.244E+00	1.933E+01	2.038E+00	1.436E-08	1.078E+00	3.520E+00	9.326E-14	6.392E+00
	Worst	8.882E-16	4.810E+00	1.994E+01	4.849E+00	3.297E+00	1.537E+00	5.447E+00	1.359E-13	1.001E+01
	Mean	8.882E-16	3.771E+00	1.979E+01	2.891E+00	1.969E+00	1.344E+00	4.283E+00	1.105E-13	7.696E+00
	S. Deviation	0.000E+00	3.755E-01	2.030E-01	6.188E-01	6.592E-01	1.383E-01	4.353E-01	1.007E-14	9.098E-01
F11	Best	0.000E+00	1.995E+00	1.457E+01	7.577E+01	6.439E-15	1.018E+00	3.718E+00	0.000E+00	4.373E+00
	Worst	0.000E+00	4.653E+00	1.004E+02	1.407E+02	1.478E-02	1.105E+00	1.290E+01	1.305E-02	1.010E+01
	Mean	0.000E+00	2.599E+00	4.241E+01	9.876E+01	2.957E-03	1.074E+00	7.647E+00	6.522E-04	6.941E+00
	S. Deviation	0.000E+00	5.600E-01	2.424E+01	1.853E+01	4.902E-03	2.411E-02	2.442E+00	2.917E-03	1.440E+00
F12	Best	4.530E-02	2.441E+00	1.372E+03	3.294E+00	5.369E-02	1.630E+01	2.379E+00	1.780E-01	7.268E+00
	Worst	1.939E-01	1.401E+01	1.128E+06	8.943E+00	4.600E+00	4.376E+01	6.232E+00	4.148E-01	2.069E+01
	Mean	9.344E-02	6.613E+00	2.655E+05	4.901E+00	1.571E+00	2.591E+01	3.814E+00	2.642E-01	1.265E+01
	S. Deviation	3.797E-02	2.464E+00	3.345E+05	1.416E+00	1.350E+00	7.307E+00	1.055E+00	5.458E-02	3.638E+00
F13	Best	1.031E-01	1.245E+02	1.466E+05	8.801E+01	1.084E-01	1.369E+02	5.219E+01	5.767E+00	1.315E+02
	Worst	1.947E+00	5.621E+02	1.252E+07	3.489E+02	8.077E+00	5.095E+03	1.416E+02	6.827E+00	9.108E+03
	Mean	1.418E+00	2.277E+02	2.403E+06	1.455E+02	2.518E+00	9.882E+02	1.004E+02	6.284E+00	9.780E+02
	S. Deviation	2.868E-01	9.803E+01	2.987E+06	5.668E+01	1.720E+00	1.078E+03	2.583E+01	3.174E-01	2.018E+03

Table 9: Comparison results of the first 13 benchmark functions (F14-F23) with fixed-dimension, where the best performance for each algorithm is highlighted in red

Function		Different Algorithms								
		TLCO	PSO	GA	GSA	FA	DE	ASO	GWO	CS
F14	Best	0.998	0.998	0.998	5.929	0.998	0.998	1.992	0.998	0.998
	Worst	0.998	11.719	14.563	22.901	0.998	7.874	12.936	12.671	0.998
	Mean	0.998	3.898	4.724	15.159	0.998	1.342	6.943	4.527	0.998
	S. Deviation	3.222E-16	3.301E+00	3.946E+00	5.688E+00	2.926E-16	1.538E+00	3.609E+00	4.026E+00	7.204E-17
F15	Best	0.000308	0.000307	0.000592	0.002028	0.000307	0.000398	0.000695	0.000307	0.000307
	Worst	0.001	0.020	0.020	0.024	0.001	0.001	0.002	0.020	0.001
	Mean	0.000	0.001	0.003	0.010	0.000	0.001	0.001	0.005	0.000
	S. Deviation	2.251E-04	4.470E-03	6.019E-03	9.731E-03	2.817E-04	1.762E-04	2.445E-04	8.881E-03	9.131E-05
F16	Best	-1.031600	-1.031600	-1.031600	-1.031600	-1.031600	-1.031600	-1.031600	-1.031600	-1.031600
	Worst	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032
	Mean	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032	-1.032
	S. Deviation	1.441E-16	2.278E-16	2.278E-16	8.823E-17	5.094E-17	2.278E-16	2.161E-16	6.541E-09	1.837E-16
F17	Best	0.397890	0.397890	0.397890	0.397890	0.397890	0.397890	0.397890	0.397890	0.397890
	Worst	0.398	0.398	0.398	0.398	0.398	0.398	0.398	0.398	0.398
	Mean	0.398	0.398	0.398	0.398	0.398	0.398	0.398	0.398	0.398
	S. Deviation	7.945E-16	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	1.398E-06	8.514E-09
F18	Best	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000
	Worst	3.000	3.000	30.000	3.000	3.000	3.000	3.000	84.000	3.000

	Mean	3.000	3.000	4.350	3.000	3.000	3.000	3.000	7.050	3.000
	S. Deviation	3.007E-15	5.391E-16	6.037E+00	1.098E-14	6.681E-16	2.038E-16	1.805E-15	1.811E+01	9.772E-16
F19	Best	-3.862800	-3.862800	-3.862800	-3.862800	-3.862800	-3.862800	-3.862800	-3.862800	-3.862800
	Worst	-3.863	-3.863	-3.863	-3.863	-3.863	-3.863	-3.557	-3.855	-3.863
	Mean	-3.863	-3.863	-3.863	-3.863	-3.863	-3.863	-3.848	-3.862	-3.863
	S. Deviation	1.920E-15	2.278E-15	2.278E-15	1.726E-15	1.699E-15	2.278E-15	6.845E-02	2.675E-03	6.612E-07
F20	Best	-3.322000	-3.322000	-3.322000	-3.322000	-3.322000	-3.322000	-3.322000	-3.322000	-3.322000
	Worst	-3.138	-3.203	-3.203	-3.322	-3.203	-3.319	-3.203	-3.080	-3.322
	Mean	-3.271	-3.292	-3.257	-3.322	-3.269	-3.322	-3.316	-3.277	-3.322
	S. Deviation	6.605E-02	5.282E-02	6.069E-02	2.278E-16	6.069E-02	6.605E-04	2.659E-02	7.540E-02	2.437E-05
F21	Best	-10.153000	-10.153000	-10.153000	-10.153000	-10.153000	-10.153000	-10.153000	-10.153000	-10.153000
	Worst	-8.865	-2.631	-2.631	-2.631	-2.683	-5.097	-2.683	-5.055	-5.101
	Mean	-9.957	-5.395	-5.271	-7.536	-9.272	-9.895	-6.907	-9.140	-7.122
	S. Deviation	3.298E-01	3.332E+00	3.389E+00	3.659E+00	2.198E+00	1.130E+00	3.716E+00	2.078E+00	2.539E+00
F22	Best	-10.403000	-10.403000	-10.403000	-10.403000	-10.403000	-10.403000	-10.403000	-10.403000	-10.403000
	Worst	-1.838	-1.838	-2.752	-10.403	-3.724	-10.300	-2.752	-5.088	-5.129
	Mean	-9.367	-5.126	-8.014	-10.403	-10.069	-10.397	-8.785	-10.137	-8.821
	S. Deviation	2.407E+00	3.250E+00	3.487E+00	2.734E-15	1.493E+00	2.309E-02	2.760E+00	1.188E+00	2.480E+00
F23	Best	-10.536000	-10.536000	-10.536000	-10.536000	-10.536000	-10.536000	-10.536000	-10.536000	-10.536000
	Worst	-3.835	-2.422	-2.422	-3.835	-2.871	-10.536	-2.871	-5.129	-5.176
	Mean	-9.917	-6.504	-7.813	-10.201	-10.153	-10.536	-9.003	-10.265	-7.856
	S. Deviation	1.871E+00	3.795E+00	3.820E+00	1.498E+00	1.714E+00	1.482E-09	3.146E+00	1.209E+00	2.750E+00

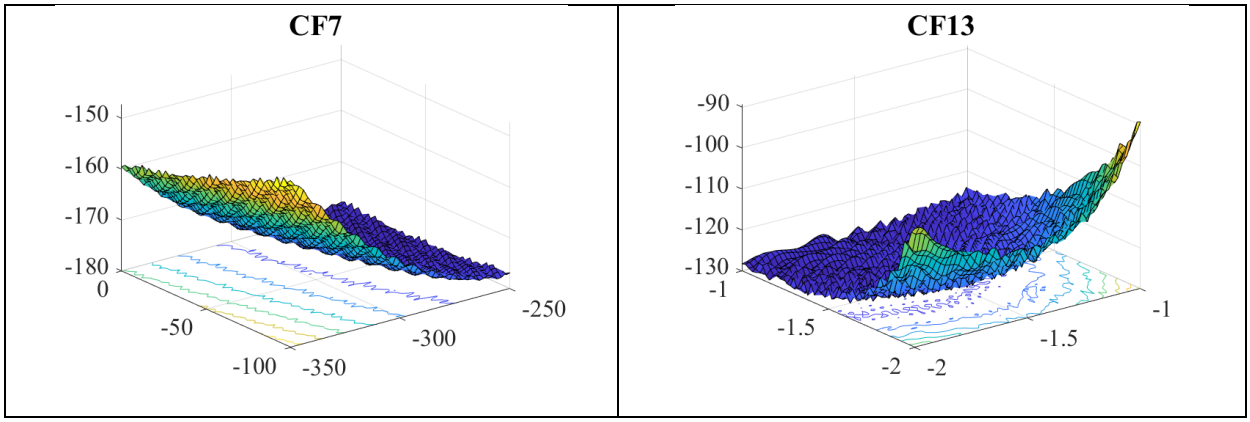
5 CEC 2005 benchmark functions

In this section, we check the effectiveness of the TLCO algorithm in solving high-complex problems. Seven benchmark functions representing different properties in CEC 2005 [80] are used to evaluate the performance of TLCO. The properties of these functions range from simple to complex; they are summarized in **Erreur ! Référence non valide pour un signet.**, and **Fig. 11** shows a 3D representation.

Table 10: Descriptions of 7 benchmark functions in CEC2005

Functions	Description	Properties	Dimension	Solution space
Multimodal functions				
CF7	Shifted Rotated Griewank's Function without Bounds	M, R, S^*, N, S, N^*		
Expanded functions				
CF13	Shifted Expanded Griewank's plus Rosenbrock's Function	M, S, N, S^*		
CF14	shifted Rotated Expanded Scaffer's CF6 Function	M, S, N, S^*		
Hybrid composite functions				
CF16	Rotated Version of Hybrid Composition Function CF15	M, R, S, A, D, S^{**}		
CF18	Rotated Hybrid Composition Function	M, R, N, S		
CF20	Rotated Hybrid Composition Function with Global Optimum on the Bounds	M, N, S, A, D, S^{**}, A^*, G	D	$[-5, 5]$
CF25	Rotated Hybrid Composition Function without bounds	M, N, S, A, D, U, G, N^*	D	$[2, 5]$

Note that: M: *multi-modal*, N: *non-separable*, S: *scalable*, S^*: *Shifted, R: rotated,* A: *a huge number of local optima*, D: *different function's properties are mixed together*, S^{**}: *sphere Functions give two flat areas for the function*, A^*: *a local optimum is set on the origin*, G: *global optimum is on the bound*, N^*: *non-continuous*, U: *uni-modal functions give flat areas for the function*, N^* : *no bounds.*



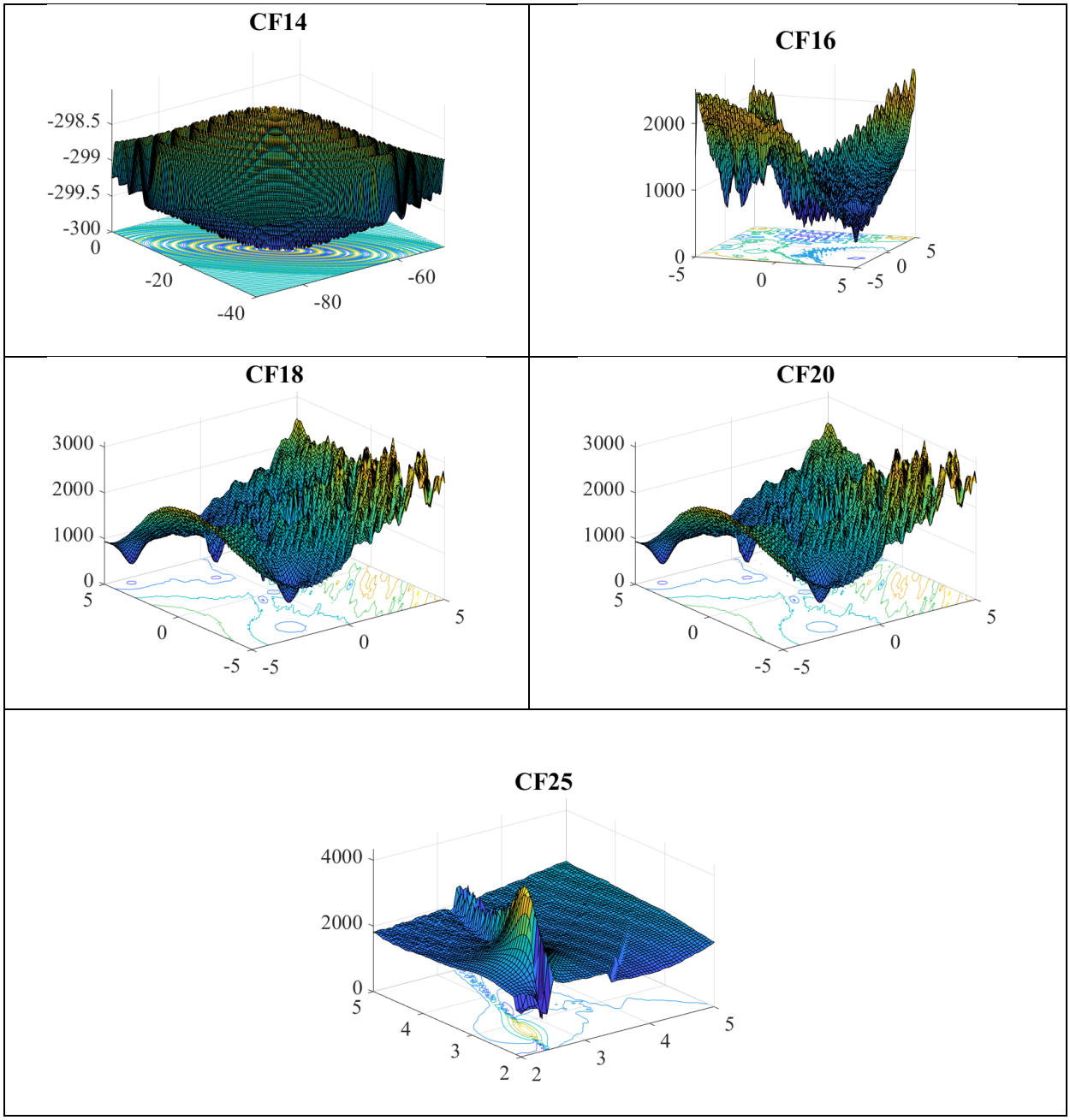


Fig. 11: 3D visualization of functions CF20-CF25

These benchmark functions are really high level 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. 12** shows the ability of TLCO to escape from local optima for the functions CF16 with dimension $D = 2$. TLCO fails to reach global optimal with 30, 50 and 100 iterations as shown in **Fig. 12a, Fig. 12b, Fig. 12c**. However, with the increasing of the iterations, this ability of TLCO is improved and TLCO can find the best global with acceptance error with 500 iterations as shown in **Fig. 12d.**

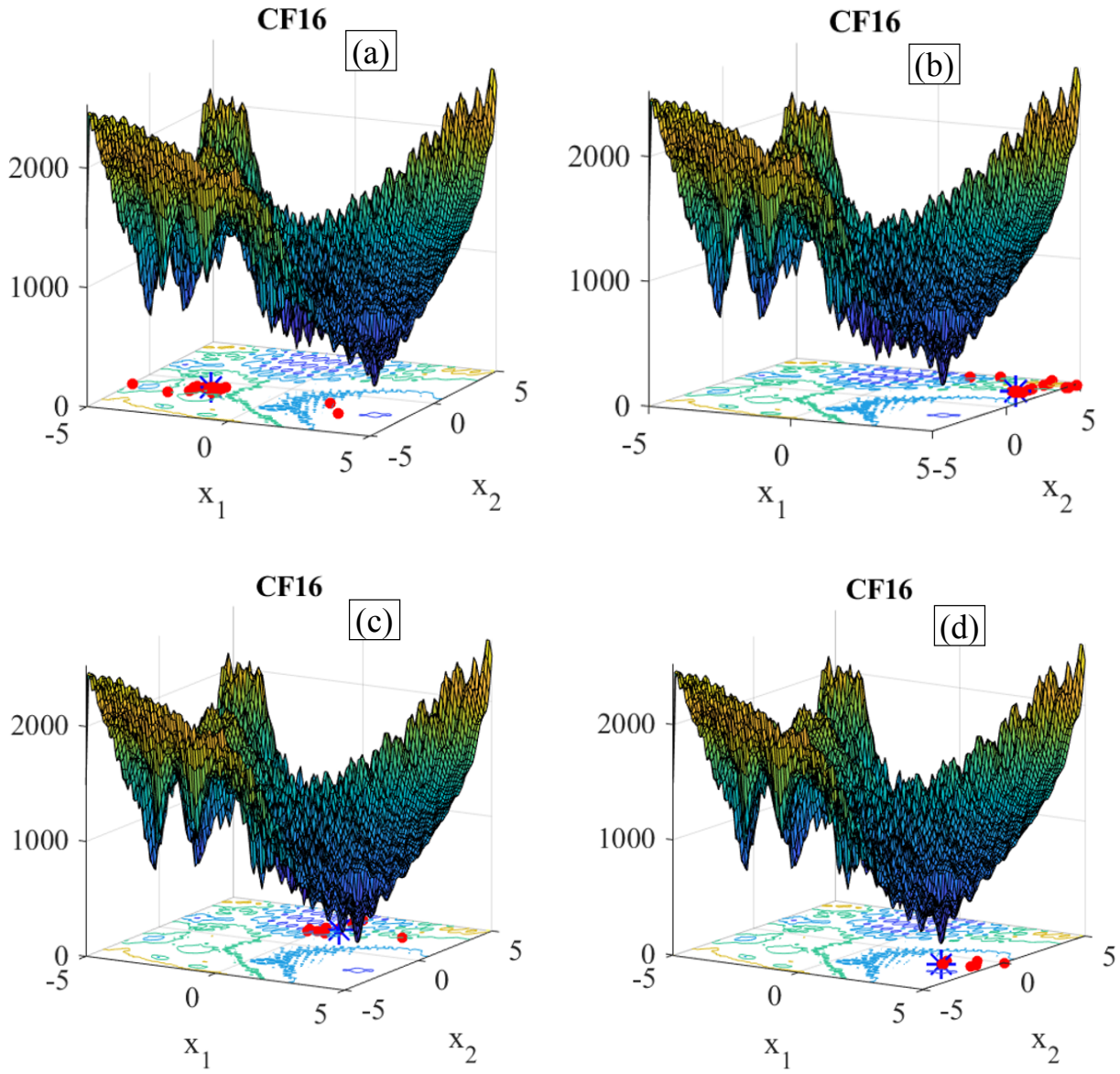


Fig. 12: The ability of escaping local optimal of TLCO in different iteration: (a) with 30 iteration, (b) with 50 iteration, (c) with 100 iteration, (d) with 500 iterations

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 = 100, the number of particles are N = 30 and the total number of iterations (1000). Four metrics including best value, the worst value, the mean value and the standard deviation are computed for each algorithm with 50 independent runs as shown in **Erreur ! Référence non valide pour un signet.**. Once again, TLCO still proves its reliability and effectiveness in solving high complexity functions. TLCO still achieves the best performance in the case of functions CF7, CF18, CF20 and CF25 in comparison with other algorithms.

Table 11: Comparison results of 7 CEC 2005 benchmark functions where the best performance for each algorithm is highlighted in red

Function		Different Algorithms								
		TLCO	PSO	GA	GSA	FA	DE	ASO	GWO	CS
CF7	Best	4516.289	4516.289	4516.289	9867.524	4516.289	4516.289	4516.289	4516.331	10622.544
	Worst	4516.289	4571.513	4516.289	12043.814	4516.289	4544.741	4544.741	4516.358	12233.661
	Mean	4516.289	4527.333	4516.289	11305.698	4516.289	4521.979	4521.979	4516.346	11588.736
	S. Deviation	6.431E-13	2.470E+01	6.431E-13	9.106E+02	1.676E-09	1.272E+01	1.272E+01	1.214E-02	6.340E+02
CF13	Best	-126.848	-127.549	-127.291	-124.866	-127.880	-122.083	-127.509	-125.984	-121.192
	Worst	-123.927	-124.175	-123.543	-123.616	-124.907	-119.581	-125.962	-122.652	-116.985
	Mean	-124.977	-126.199	-124.796	-124.347	-126.687	-120.870	-126.744	-124.511	-119.397
	S. Deviation	1.120E+00	1.403E+00	1.583E+00	4.534E-01	1.112E+00	9.257E-01	7.149E-01	1.291E+00	1.844E+00
CF14	Best	-287.395	-287.655	-286.419	-286.489	-290.228	-286.499	-286.363	-288.180	-287.333
	Worst	-286.523	-286.882	-285.893	-285.822	-287.988	-286.306	-285.884	-287.294	-286.843
	Mean	-286.940	-287.290	-286.206	-286.165	-289.114	-286.402	-286.126	-287.799	-287.108
	S. Deviation	3.521E-01	3.351E-01	2.661E-01	2.684E-01	9.620E-01	7.899E-02	1.753E-01	3.356E-01	1.794E-01
CF16	Best	246.587	250.426	274.287	442.789	170.555	369.153	169.764	200.225	264.791
	Worst	534.907	674.587	521.204	620.000	196.831	441.102	666.821	673.053	307.399
	Mean	374.691	586.873	356.006	539.980	186.054	406.019	542.575	397.424	287.827
	S. Deviation	1.049E+02	1.881E+02	9.881E+01	2.124E+02	9.604E+00	2.974E+01	2.133E+02	2.125E+02	1.783E+01
CF18	Best	910.000	922.563	913.824	1059.984	913.916	917.425	910.000	945.205	919.390
	Worst	910.000	926.631	918.241	1170.991	917.741	918.556	910.000	961.522	921.142
	Mean	910.000	925.099	915.301	1065.116	915.764	918.037	910.000	953.343	920.287
	S. Deviation	8.212E+00	1.542E+00	1.741E+00	4.615E+00	1.630E+00	4.995E-01	1.150E-12	5.803E+00	6.732E-01
CF20	Best	910.000	923.640	916.519	1051.377	914.884	916.467	910.000	950.380	918.365
	Worst	910.000	1135.349	926.263	1169.942	922.263	918.174	910.000	975.032	920.530
	Mean	910.000	996.210	922.505	1067.591	917.751	917.531	910.000	961.727	919.631
	S. Deviation	0.000E+00	8.627E+01	3.922E+00	1.924E+00	2.745E+00	6.504E-01	1.189E-11	1.074E+01	8.400E-01
CF25	Best	1871.753	1936.635	1960.826	1915.139	1947.123	1887.181	1875.231	1893.870	2103.599
	Worst	1891.198	2009.479	1984.600	1964.213	1957.460	1892.065	1893.584	1906.951	2165.611
	Mean	1884.510	1973.677	1971.156	1830.685	1852.531	1889.341	1884.694	1898.332	2141.866
	S. Deviation	3.917E+00	3.348E+01	9.167E+00	1.331E+02	4.957E+00	2.000E+00	8.217E+00	5.557E+00	2.343E+01

6 Engineering design problems

This section illustrates the reliability of TLCO to solve real-world problems, in the case of 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 30 particles and 2000 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.

6.1 Tension/compression spring

Solving this problem requires to find the minimum weight of spring whose three changeable d, D and P including wire diameter (d), mean coil diameter (D), and the number of active coils (P) as shown in **Fig. 13**. 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:

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

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

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

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

Where

$$0.05 \le x_1 \le 2,\ 0.25 \le x_2 \le 1.3,\ 2 \le x_3 \le 15$$

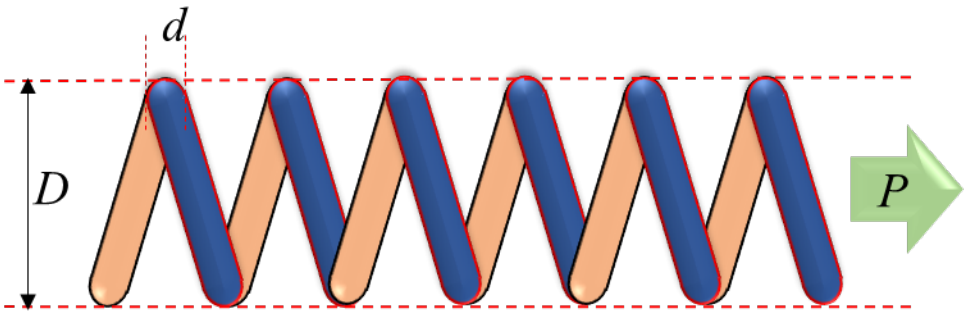


Fig. 13: Tension/compression spring design problem where the design variables are active coils (P), mean coil diameter (D), and wire diameter (d)

Erreur ! Référence non valide pour un signet. shows the best result obtained using TLCO with the corresponding values of constrained functions from $g_1(x)$ to $g_4(x)$. The comparison results between TLCO and other methods including 11 well-known optimization algorithms, and mathematical technique, which are published in the literature as shown in **Table 13**. CSA [81] gets the best performance with ***f(x)*** = *0.01266523*. This result is still exploited again by TLCO with ***f(x)*** = *0.01266523447265*. It can be seen that the best optimal result using TLCO is very competitive in comparison with those reported by CSA and better than the remaining ones.

Table 12: The best result obtained obtained using TLCO for tension/compression spring design

Variables	x_1	x_2	x_3	
	0.05169867161112	*0.35694898786358*	*11.275421263651*	
The value of constrained functions	g_1	g_2	g_3	g_4
	0.00E+00	*0.00E+00*	*-4.047044892*	*-0.72756822*
The best result ***f(x)***	*0.01266523447265*			

Table 13: 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 [82]	0.05172800	0.35764400	11.24454300	0.01267470
ES [83]	0.355360	0.051643	11.397926	0.012698
GA [84]	0.05148000	0.35166100	11.63220100	0.01270480
GWO [59]	0.05169	0.356737	11.28885	0.0126660
BMO [55]	0.05165974	0.35601249	11.33044295	0.012665264
ABC [85]	0.05174900	0.35817900	11.20376300	0.01266500
RO [86]	0.051370	0.349096	11.762790	0.012679
CSA [81]	0.05168903	0.35671695	11.28901180	0.01266523
WOA [62]	0.051207	0.345215	12.004032	0.012676
DE [87]	0.05160900	0.35471400	11.41083100	0.01267020
HS [88]	0.051154	0.349871	12.076432	0.012671
Constraint	0.05000000	0.31590000	14.25000000	0.01283340
In this research				
TLCO	*0.05169867161112*	*0.35694898786358*	*11.275421263651*	*0.01266523447265*

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 **The best** result gained by TLCO as shown in **Erreur ! Référence non valide pour un signet.; Table 15** shows its comparison with other optimization methods found in the literature. The current best result belong to BMO [55] with ***f(x)*** = *5887.097014;* The best optimal result obtained with other algorithms fluctuate around ***f(x)*** = *6059.* Here again TLCO appears to be the best algorithm in comparison with other algorithms with the best results reported for ***f(x)*** = *5885.3327736165.* It is streets ahead of all the other algorithms on this problem. 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 *S* which is short enough during the last few iterations to enhance the level of accuracy. According to our statistics, the results reported in **Erreur ! Référence** non valide pour un signet. may be regarded as a new record for solving this problem.

Table 14. There are four linear and nonlinear constraints affecting the design of pressure vessel. T_s, T_h are the thickness of shell and the thickness of the head, respectively. The inner radius R, and the length of the cylindrical section without considering the head L.

The mathematical formulation of this problem was 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

$$g_1(x) = -x_1 + 0.0193x_3 \le 0$$
$$g_2(x) = -x_2 + 0.00954x_3 \le 0$$
$$g_3(x) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0$$
$$g_4(x) = x_4 - 240 \le 0$$

Where

$$0.0625 \le x_1 \le 99 \times 0.0625, \quad 0.0625 \le x_2 \le 99 \times 0.0625, \quad 10 \le x_3 \le 200, \quad 10 \le x_4 \le 200$$

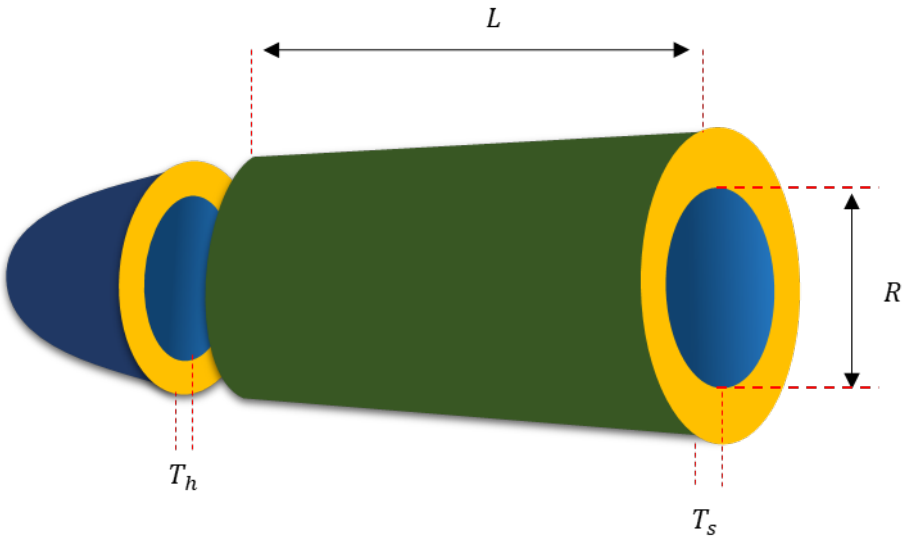


Fig. 14: Pressure vessel design problem where the design variables are inner radius *R*, the length *L*, thickness shell T_s and thickness of the head T_h

The best result gained by TLCO as shown in **Erreur ! Référence non valide pour un signet.**; **Table 15** shows its comparison with other optimization methods found in the literature. The current best result belong to BMO [55] with ***f(x)*** = *5887.097014*; The best optimal result obtained with other algorithms fluctuate around ***f(x)*** = *6059*. Here again TLCO appears to be the best algorithm in comparison with other algorithms with the best results reported for ***f(x)*** = *5885.3327736165*. It is streets ahead of all the other algorithms on this problem. 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 *S* which is short enough during the last few iterations to enhance the level of accuracy. According to our statistics, the results reported in **Erreur ! Référence non valide pour un signet.** may be regarded as a new record for solving this problem.

Table 14: The best result obtained using TLCO for pressure vessel

Variables	x_1	x_2	x_3	x_4
	0.77816864137511	*0.384649162627902*	*40.3196187240987*	*200*
The value of constrained functions	g_1	g_2	g_3	g_4
	-1.1102E-16	*-2.22E-16*	*-2.33E-10*	*-40*
The best result ***f(x)***	*5885.3327736165*			

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 [82]	0.812500	0.437500	42.091266	176.7465	6061.0777
ES [83]	0.812500	0.437500	42.098087	176.640518	6059.745605
GA [84]	0.812500	0.437500	40.3239	200.000000	6288.7445
GWO [59]	0.812500	0.434500	42.089181	176.758731	6051.5639

ABC [85]	0.812500	0.437500	42.098446	176.636596	6059.714339
CSA [81]	0.812500	0.437500	42.09844539	176.6365986	6059.714363
WOA [62]	0.812500	0.437500	42.0982699	176.638998	6059.7410
HS [88]	1.125000	0.625000	58.29015	43.69268	7197.730
DE [87]	0.812500	0.437500	42.098411	176.637690	6059.7340
ACO [90]	0.812500	0.437500	42.103624	176.572656	6059.0888
MVO [91]	0.8125	0.4375	42.090738	176.73869	6060.8066
BMO [55]	0.7789243362	0.3850096372	40.35569043	199.5028780967	5887.097014
In this research					
TLCO(*)	*0.778168641*	*0.384649162*	*40.319618724*	*200*	*5885.3327736165*

Note that: subscript (*) is the best solution for pressure vessel problem register in the literature

6.3 Welded beam design

The objective is to minimize the total cost of a welded beam as given in **Fig. 15**. 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: $f(x) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2)$

Subject to:

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

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

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

$$g_4(x) = 1.1047 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2) - 5 \le 0$$

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

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

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

Where

$$\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \tau' = \frac{P}{\sqrt{2}x_1 x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right)$$

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

$$P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right), P = 6000\ lb, L = 14\ in, E = 30\times10^6\ psi, G = 12\times10^6\ psi$$

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

$$0.1 \le x_1 \le 2.0 \qquad 0.1 \le x_2 \le 10 \qquad 0.1 \le x_4 \le 2.0$$

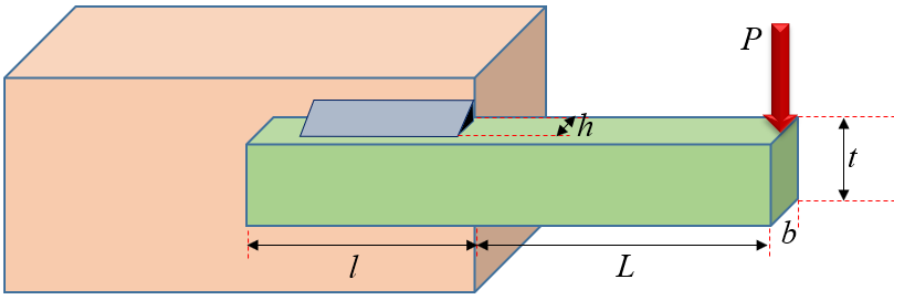


Fig. 15: Structure of Welded beam design. There are four optimization variables including the thickness of weld (h), the length of weld (l), the thickness of bar (b) and the height of bar (t). This problem is designed with 7 constraint equations whose variables are shear stress (τ), bending stress in beam (θ), buckling load on bar (P_c) and the deflection (δ).

The best results obtained with TLCO for solving this problem are given in **Erreur ! Référence non valide pour un signet.**. And the comparison between TLCO and other algorithms is shown in **Table 17.** It can be noticed that almost algorithms can find the best results with an acceptable error except HS [79]. The best results in this problem are found to be around *f(x) = 1.72*. The best total cost exploited by TLCO is still better in comparison with other algorithms and a litter bit lower than that of ACO [90] which gives the best performance.

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

Variables	x_1	x_2	x_3	x_4
	0.20570987476921	3.470985710499610	9.0364379313505	0.205738108120211
The value of constrained functions	g_1	g_2	g_3	g_4
	-3.6380E-12	-2.22E-16	-2.33E-10	-40
	g_5	g_6	g_7	
	-8.0710E-02	-2.36E-01	-6.60E-01	
The best result $f(x)$	1.7249209835			

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 [90]	0.2057	3.471131	9.036683	0.205731	1.724918
GA [92]	0.205986	3.471328	9.020224	0.20648	1.728226
ES [93]	0.20573	3.470489	9.036624	0.205729	1.724852
ABC [85]	0.20573	3.470489	9.036624	0.20573	1.724852
DE [87]	0.203137	3.542998	9.033498	0.206179	1.733462
PSO [82]	0.202369	3.544214	9.04821	0.205723	1.728024
GWO [59]	0.205676	3.478377	9.03681	0.205778	1.72624
MVO [91]	0.205463	3.473193	9.044502	0.205695	1.72645
RO [86]	0.203687	3.528467	9.004233	0.207241	1.735344
HS [88]	0.2442	6.2231	8.2915	0.2400	2.3807
DE [87]	0.203137	3.542998	9.033498	0.206179	1.733462
In this research					
TLCO	*0.20570987*	*3.47098571*	*9.03643793*	*0.20573810*	*1.72492098*

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. 16**. 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).

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:

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

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

$$g_5(x) = \frac{1}{110x_6^3}\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9\times10^6} - 1 \le 0$$

$$g_6(x) = \frac{1}{85x_7^3}\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5\times10^6} - 1 \le 0$$

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

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

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

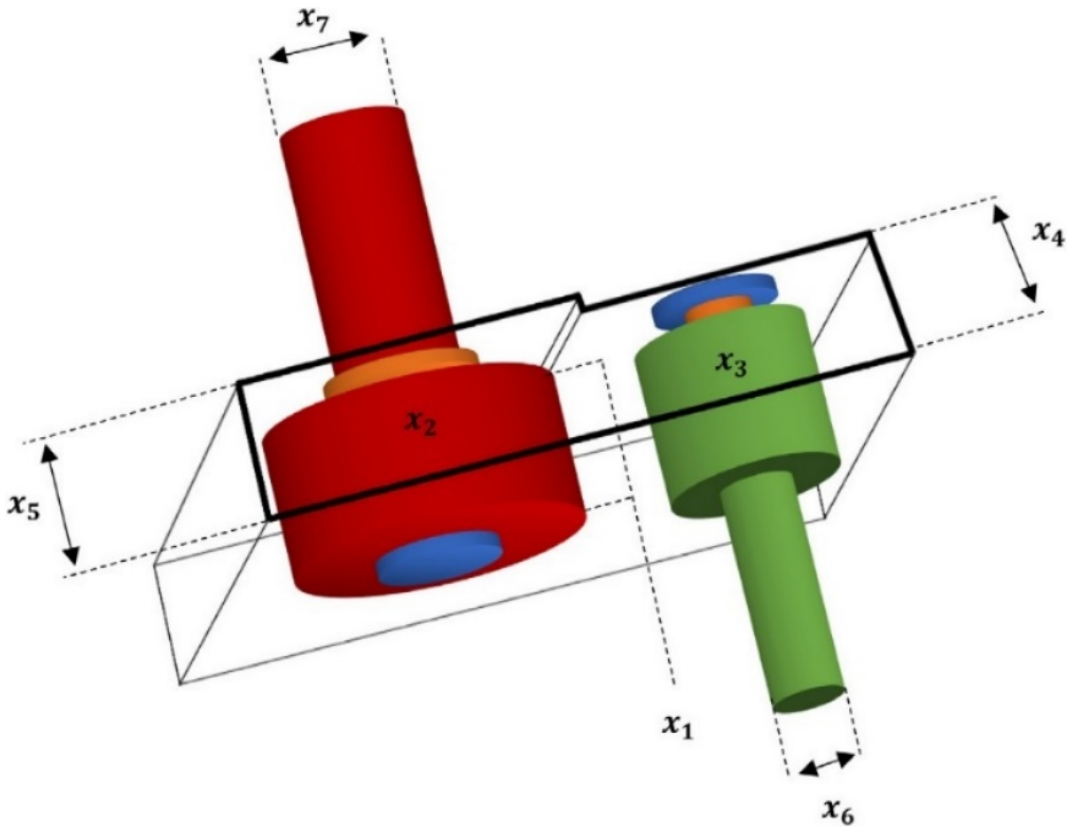


Fig. 16: A schematic of the speed reducer design

Table 18 and **Table 19**, respectively, the best result achieved by TLCO and its comparison with other algorithms. According to the statistical figures for this problem, PSO-DE [85] has the best performance with *f(x) = 2996.348100* while the best results of the other algorithms fluctuated between *2996* to *3010*. FA [86] 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 level of accuracy. The best value exploited by TLCO streets 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^{-15} as given in condition of $g_5(x)$ 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 where current algorithms cannot approach because their movement strategy is not yet perfect.

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

Variables	x_1	x_2	x_3
	3.50000000000001	*0.7*	*17*

	x_4	x_5	x_6	x_7
	7.3	7.71531991150231	3.35021466609744	5.28665446498023
The value of constrained functions	g_1	g_2	g_3	
	-0.07391528	-0.197998527	-0.499172248	
	g_4	g_5	g_6	g_7
	-0.904643905	-8.87E-13	-1.33E-15	-0.7025
	g_8	g_9	g_{10}	g_{11}
	-1.89E-15	-0.583333333	-0.051325754	-3.12E-12
The best result $f(x)$	2994.47106614761			

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 [94]	3.499000	0.699900	17.000000	7.300000	7.800000	3.350200	5.287200	2996.783000
GWO [59]	3.501000	0.700000	17.000000	7.300000	7.811013	3.350704	5.287411	2997.819650
CS [50]	3.501500	0.700000	17.000000	7.605000	7.818100	3.352000	5.287500	3000.981000
WSA [95]	3.500000	0.700000	17.000000	7.300000	7.800000	3.350215	5.286683	2996.348225
MFO [96]	3.497455	0.700000	17.000000	7.827750	7.712457	3.351787	5.286352	2998.940830
SCA [78]	3.521000	0.700000	17.000000	8.300000	7.923351	3.355911	5.300734	3026.837720
AOA [97]	3.503840	0.700000	17.000000	7.300000	7.729330	3.356490	5.286700	2997.915700
LGSI4 [95]	3.501000	0.700000	17.000000	7.300000	7.800000	3.350214	5.286683	2996.348205
PSO-DE[98]	3.500000	0.700000	17.000000	7.300000	7.800000	3.350210	5.286680	2996.348100
LGSI2 [95]	3.500000	0.700000	17.000000	7.300000	7.800000	3.350215	5.286683	2996.348166
FA [99]	3.507495	0.700100	17.000000	7.719674	8.080854	3.351512	5.287051	3010.137492
In this research								
TLCO(*)	*3.50000*	*0.70000000*	*17.00000000*	*7.30000000*	*7.71531991*	*3.35021467*	*5.28665446*	*2994.47106*

Note that: subscript (*) is the best solution for speed reducer design register in the literature

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. 17**. 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 element, thus, 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, in each of 16 groups have the same value as following:

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:

$\sigma_{min} \le \sigma_i \le \sigma_{max}, \ i = 1,2,...,72$

$\delta_{min} \le \delta_i \le \delta_{max}, \ i = 1,2,...,72$

Where

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

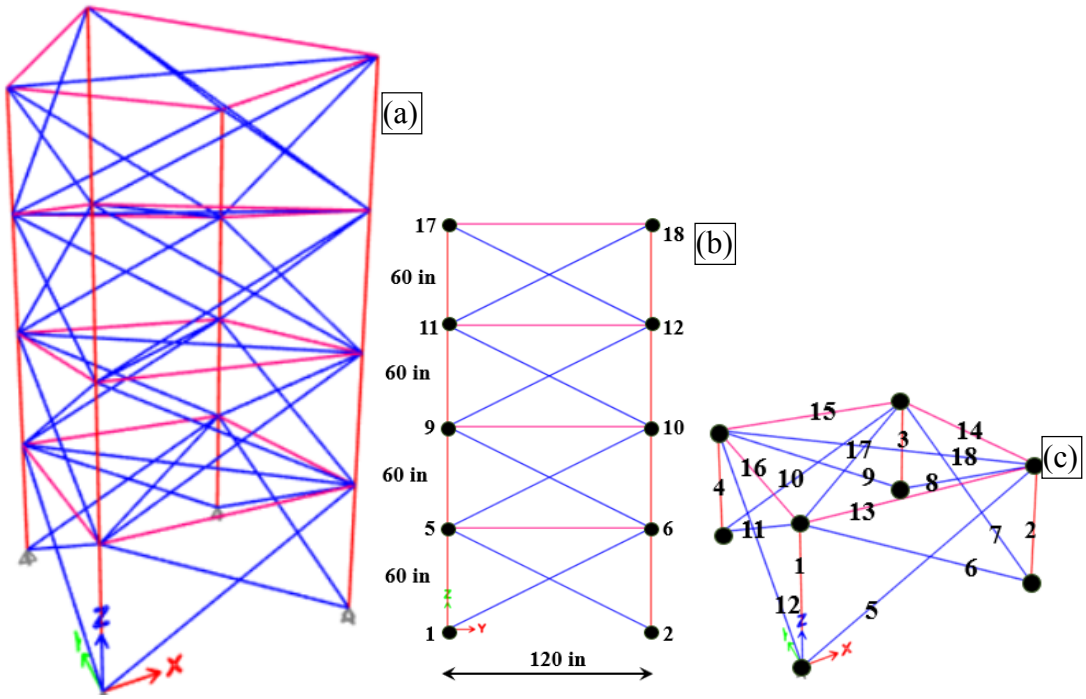


Fig. 17: 72-bar space truss design problem: (a) 3D model, (b) dimension and node, (c) element numbering pattern for first story.

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

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 shows that GGP [100] gets the best performance with *f(x) = 379.31*. TLCO and HBB-BC [101] both rank 2nd 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 [91]

and still performs better than other algorithms. Especially, TLCO demonstrates its superiority compared to three algorithms including GA [102], ACO [103] and PSO [104].

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

Groups	Members	TLCO	GGP [100]	GA [102]	ACO [103]	PSO [104]	BB-BC [105]	HBB-BC [101]
1	1–4	*1.8911732*	2.0259	1.8562	1.9480	1.7427	1.8577	1.9042
2	5–12	*0.5128193*	0.5332	0.4933	0.5080	0.5185	0.5059	0.5162
3	13–16	*0.1000000*	0.1000	0.1000	0.1010	0.1000	0.1000	0.1000
4	17,18	*0.1000000*	0.1000	0.1000	0.1020	0.1000	0.1000	0.1000
5	19–22	*1.2740327*	1.1567	1.2830	1.3030	1.3079	1.2476	1.2582
6	23–30	*0.5206884*	0.5689	0.5028	0.5110	0.5193	0.5269	0.5035
7	31–34	*0.1000000*	0.1000	0.1000	0.1010	0.1000	0.1000	0.1000
8	35,36	*0.1000000*	0.1000	0.1000	0.1000	0.1000	0.1012	0.1000
9	37–40	*0.5118870*	0.5137	0.5177	0.5610	0.5142	0.5209	0.5178
10	41–48	*0.5151597*	0.4791	0.5227	0.4920	0.5464	0.5172	0.5214
11	49–52	*0.1000000*	0.1000	0.1000	0.1000	0.1000	0.1004	0.1000
12	53,54	*0.1000008*	0.1000	0.1049	0.1070	0.1095	0.1005	0.1007
13	55–58	*0.1569718*	0.1579	0.1557	0.1560	0.1615	0.1565	0.1566
14	59–66	*0.5365165*	0.5501	0.5501	0.5500	0.5092	0.5507	0.5421
15	67–70	*0.4183381*	0.3449	0.3981	0.3900	0.4967	0.3922	0.4132
16	71,72	*0.5649648*	0.4984	0.6749	0.5920	0.5619	0.5922	0.5756
Weight		*379.661117*	379.31	380.32	380.24	381.91	379.85	379.66

7 CONCLUSION

The 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 and soldiers. Thus, the termite workers in TLCO perform the exploration and the soldiers ensure the exploitation. This ensures at each iteration a balance between the exploration and the exploitation. Besides, a connection and interdependence between workers and soldiers is created by shared information of the best solution found at the previous iteration. Thus, when the best solution value is updated, this information is transmitted to each worker and soldier in the next iteration to adjust its movements. As a result, the movement strategy in TLCO is always clearly oriented and termite soldiers reach the potential search space around the best solution, Meanwhile, termite workers are secured for expanding the new search space.

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 few first 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 uses parameters to create two random movement directions, these parameters being bounded in [-1, 1] and it can randomly receive negative or positive values during the process of position updating. If the parameters are 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 from 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 present. With large enough iterations, TLCO can still find the best value. Five engineering problems were also used to evaluate the reliability and the effectiveness of TLCO. The statistical results show that TLCO is the best algorithm in 3 engineering problems, Especially, for pressure vessel design problem and speed reducer design problem, the results reported using TLCO can be considered as the 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 highly reliable in solving optimization problems. It can be seen as a robust algorithm for solving real problems in many fields.

References.

1. Haupt, R.L. and S. Ellen Haupt, *Practical genetic algorithms.* 2004.
2. Boussaïd, I., J. Lepagnot, and P. Siarry, *A survey on optimization metaheuristics.* Information sciences, 2013. **237**: p. 82-117.
3. Mahdavi, S., M.E. Shiri, and S. Rahnamayan, *Metaheuristics in large-scale global continues optimization: A survey.* Information Sciences, 2015. **295**: p. 407-428.
4. Poorzahedy, H. and O.M. Rouhani, *Hybrid meta-heuristic algorithms for solving network design problem.* European Journal of Operational Research, 2007. **182**(2): p. 578-596.
5. Fister Jr, I., et al., *A brief review of nature-inspired algorithms for optimization.* arXiv preprint arXiv:1307.4186, 2013.
6. Mühlenbein, H., M. Gorges-Schleuter, and O. Krämer, *Evolution algorithms in combinatorial optimization.* Parallel computing, 1988. **7**(1): p. 65-85.
7. Gong, D., J. Sun, and X. Ji, *Evolutionary algorithms with preference polyhedron for interval multi-objective optimization problems.* Information Sciences, 2013. **233**: p. 141-161.
8. Biswas, A., et al., *Physics-inspired optimization algorithms: a survey.* Journal of Optimization, 2013. **2013**.

9. Krause, J., et al., *A survey of swarm algorithms applied to discrete optimization problems*, in *Swarm Intelligence and Bio-Inspired Computation*. 2013, Elsevier. p. 169-191.
10. Passino, K.M., *Biomimicry of bacterial foraging for distributed optimization and control.* IEEE control systems magazine, 2002. **22**(3): p. 52-67.
11. De Falco, I., et al., *Biological invasion–inspired migration in distributed evolutionary algorithms.* Information Sciences, 2012. **207**: p. 50-65.
12. Goldberg, D.E. and J.H. Holland, *Genetic algorithms and machine learning.* 1988.
13. Gong, D., J. Sun, and Z. Miao, *A set-based genetic algorithm for interval many-objective optimization problems.* IEEE Transactions on Evolutionary Computation, 2016. **22**(1): p. 47-60.
14. Tang, K.-S., et al., *Genetic algorithms and their applications.* IEEE signal processing magazine, 1996. **13**(6): p. 22-37.
15. Grefenstette, J.J., *Genetic algorithms and their applications: proceedings of the second international conference on genetic algorithms*. 2013: Psychology Press.
16. Beyer, H.-G. and H.-P. Schwefel, *Evolution strategies–a comprehensive introduction.* Natural computing, 2002. **1**(1): p. 3-52.
17. Koza, J.R. and J.R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. Vol. 1. 1992: MIT press.
18. Storn, R. and K. Price, *Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces.* Journal of Global Optimization, 1997. **11**(4): p. 341-359.
19. Juste, K., et al., *An evolutionary programming solution to the unit commitment problem.* IEEE Transactions on Power Systems, 1999. **14**(4): p. 1452-1459.
20. Simon, D., *Biogeography-Based Optimization.* IEEE Transactions on Evolutionary Computation, 2008. **12**(6): p. 702-713.
21. Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi, *Optimization by simulated annealing.* science, 1983. **220**(4598): p. 671-680.
22. Webster, B. and P.J. Bernhard, *A local search optimization algorithm based on natural principles of gravitation*. 2003.
23. Rashedi, E., H. Nezamabadi-pour, and S. Saryazdi, *GSA: A Gravitational Search Algorithm.* Information Sciences, 2009. **179**(13): p. 2232-2248.
24. Kaveh, A. and S. Talatahari, *A novel heuristic optimization method: charged system search.* Acta Mechanica, 2010. **213**(3): p. 267-289.
25. Du, H., X. Wu, and J. Zhuang. *Small-world optimization algorithm for function optimization*. in *International conference on natural computation*. 2006. Springer.
26. Formato, R., *Central force optimization: a new metaheuristic with applications in applied electromagnetics. Prog Electromagn Res 77: 425–491*. 2007.
27. Shah-Hosseini, H., *Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation.* International Journal of Computational Science and Engineering, 2011. **6**(1-2): p. 132-140.
28. Hatamlou, A., *Black hole: A new heuristic optimization approach for data clustering.* Information sciences, 2013. **222**: p. 175-184.
29. Kaveh, A. and M. Khayatazad, *A new meta-heuristic method: Ray Optimization.* Computers & Structures, 2012. **112-113**: p. 283-294.
30. Moghaddam, F.F., R.F. Moghaddam, and M. Cheriet, *Curved space optimization: a random search based on general relativity theory.* arXiv preprint arXiv:1208.2214, 2012.
31. Zhao, W., L. Wang, and Z. Zhang, *Atom search optimization and its application to solve a hydrogeologic parameter estimation problem.* Knowledge-Based Systems, 2019. **163**: p. 283-304.
32. Moussaid, M., et al., *Collective information processing and pattern formation in swarms, flocks, and crowds.* Topics in Cognitive Science, 2009. **1**(3): p. 469-497.
33. Camazine, S., et al., *Self-organization in biological systems*. 2020: Princeton university press.

34. Bonabeau, E., et al., *Swarm intelligence: from natural to artificial systems*. 1999: Oxford university press.
35. Chakraborty, A. and A.K. Kar, *Swarm intelligence: A review of algorithms.* Nature-Inspired Computing and Optimization, 2017: p. 475-494.
36. Patnaik, S., X.-S. Yang, and K. Nakamatsu, *Nature-inspired computing and optimization*. Vol. 10. 2017: Springer.
37. Hussain, K., et al., *Metaheuristic research: a comprehensive survey.* Artificial Intelligence Review, 2019. **52**(4): p. 2191-2233.
38. Kennedy, J. and R. Eberhart. *Particle swarm optimization*. in *Proceedings of ICNN'95 - International Conference on Neural Networks*. 1995.
39. Dorigo, M., V. Maniezzo, and A. Colorni, *Ant system: optimization by a colony of cooperating agents.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 1996. **26**(1): p. 29-41.
40. Basturk, B. *An artificial bee colony (ABC) algorithm for numeric function optimization*. in *IEEE Swarm Intelligence Symposium, Indianapolis, IN, USA, 2006*. 2006.
41. Roth, M., *Termite: A swarm intelligent routing algorithm for mobile wireless ad-hoc networks.* 2005.
42. Krishnanand, K. and D. Ghose, *Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions.* Swarm intelligence, 2009. **3**(2): p. 87-124.
43. Eusuff, M., K. Lansey, and F. Pasha, *Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization.* Engineering optimization, 2006. **38**(2): p. 129-154.
44. Chu, S.-C., P.-W. Tsai, and J.-S. Pan. *Cat swarm optimization*. in *Pacific Rim international conference on artificial intelligence*. 2006. Springer.
45. Pham, D.T., et al., *The bees algorithm—a novel tool for complex optimisation problems*, in *Intelligent production machines and systems*. 2006, Elsevier. p. 454-459.
46. Pinto, P.C., T.A. Runkler, and J.M. Sousa. *Wasp swarm algorithm for dynamic MAX-SAT problems*. in *International conference on adaptive and natural computing algorithms*. 2007. Springer.
47. Mucherino, A. and O. Seref. *Monkey search: a novel metaheuristic search for global optimization*. in *AIP conference proceedings*. 2007. American Institute of Physics.
48. Yang, C., X. Tu, and J. Chen. *Algorithm of marriage in honey bees optimization based on the wolf pack search*. in *The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007)*. 2007. IEEE.
49. Lu, X. and Y. Zhou. *A novel global convergence algorithm: bee collecting pollen algorithm*. in *International Conference on Intelligent Computing*. 2008. Springer.
50. Yang, X.-S. and S. Deb. *Cuckoo search via Lévy flights*. in *2009 World congress on nature & biologically inspired computing (NaBIC)*. 2009. Ieee.
51. Shiqin, Y., J. Jianjun, and Y. Guangxing. *A dolphin partner optimization*. in *2009 WRI global congress on intelligent systems*. 2009. IEEE.
52. Yang, X.-S., *A new metaheuristic bat-inspired algorithm*, in *Nature inspired cooperative strategies for optimization (NICSO 2010)*. 2010, Springer. p. 65-74.
53. Yang, X.-S., *Firefly algorithm, stochastic test functions and design optimisation.* International journal of bio-inspired computation, 2010. **2**(2): p. 78-84.
54. Oftadeh, R., M. Mahjoob, and M. Shariatpanahi, *A novel meta-heuristic optimization algorithm inspired by group hunting of animals: hunting search.* Computers & Mathematics with Applications, 2010. **60**(7): p. 2087-2098.
55. Askarzadeh, A. and A. Rezazadeh, *A new heuristic optimization algorithm for modeling of proton exchange membrane fuel cell: bird mating optimizer.* International Journal of Energy Research, 2013. **37**(10): p. 1196-1204.
56. Gandomi, A.H. and A.H. Alavi, *Krill herd: a new bio-inspired optimization algorithm.* Communications in nonlinear science and numerical simulation, 2012. **17**(12): p. 4831-4845.

57. Pan, W.-T., *A new fruit fly optimization algorithm: taking the financial distress model as an example.* Knowledge-Based Systems, 2012. **26**: p. 69-74.
58. Kaveh, A. and N. Farhoudi, *A new optimization method: Dolphin echolocation.* Advances in Engineering Software, 2013. **59**: p. 53-70.
59. Mirjalili, S., S.M. Mirjalili, and A. Lewis, *Grey wolf optimizer.* Advances in engineering software, 2014. **69**: p. 46-61.
60. Mirjalili, S., *The ant lion optimizer.* Advances in engineering software, 2015. **83**: p. 80-98.
61. Mirjalili, S., *Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems.* Neural Computing and Applications, 2016. **27**(4): p. 1053-1073.
62. Mirjalili, S. and A. Lewis, *The whale optimization algorithm.* Advances in engineering software, 2016. **95**: p. 51-67.
63. Biyanto, T.R., et al., *Killer Whale Algorithm: An Algorithm Inspired by the Life of Killer Whale.* Procedia Computer Science, 2017. **124**: p. 151-157.
64. Saremi, S., S. Mirjalili, and A. Lewis, *Grasshopper Optimisation Algorithm: Theory and application.* Advances in Engineering Software, 2017. **105**: p. 30-47.
65. Mirjalili, S., et al., *Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems.* Advances in Engineering Software, 2017. **114**: p. 163-191.
66. Harifi, S., et al., *Emperor Penguins Colony: a new metaheuristic algorithm for optimization.* Evolutionary Intelligence, 2019. **12**(2): p. 211-226.
67. Zervoudakis, K. and S. Tsafarakis, *A mayfly optimization algorithm.* Computers & Industrial Engineering, 2020. **145**: p. 106559.
68. Chou, J.-S. and D.-N. Truong, *A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean.* Applied Mathematics and Computation, 2021. **389**: p. 125535.
69. Droste, S., T. Jansen, and I. Wegener, *Upper and lower bounds for randomized search heuristics in black-box optimization.* Theory of computing systems, 2006. **39**(4): p. 525-544.
70. Wolpert, D.H. and W.G. Macready, *No free lunch theorems for optimization.* IEEE transactions on evolutionary computation, 1997. **1**(1): p. 67-82.
71. Keller, L., *Queen lifespan and colony characteristics in ants and termites.* Insectes Sociaux, 1998. **45**(3): p. 235-246.
72. Noirot, C. and J. Pasteels, *Ontogenetic development and evolution of the worker caste in termites.* Experientia, 1987. **43**(8): p. 851-860.
73. Thorne, B.L., N.L. Breisch, and M.L. Muscedere, *Evolution of eusociality and the soldier caste in termites: influence of intraspecific competition and accelerated inheritance.* Proceedings of the National Academy of Sciences, 2003. **100**(22): p. 12808-12813.
74. Korb, J. and M. Lenz, *Reproductive decision-making in the termite, Cryptotermes secundus (Kalotermitidae), under variable food conditions.* Behavioral Ecology, 2004. **15**(3): p. 390-395.
75. Garnier, S., J. Gautrais, and G. Theraulaz, *The biological principles of swarm intelligence.* Swarm intelligence, 2007. **1**(1): p. 3-31.
76. Benhamou, S., *How many animals really do the Lévy walk?* Ecology, 2007. **88**(8): p. 1962-1969.
77. Lévy, P. and P. Lévy, *Théorie de l'addition des variables aléatoires*. 1954: Gauthier-Villars.
78. Mirjalili, S., *SCA: A Sine Cosine Algorithm for solving optimization problems.* Knowledge-Based Systems, 2016. **96**: p. 120-133.
79. Le-Duc, T., Q.-H. Nguyen, and H. Nguyen-Xuan, *Balancing composite motion optimization.* Information Sciences, 2020. **520**: p. 250-270.
80. Suganthan, P.N., et al., *Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization.* KanGAL report, 2005. **2005005**(2005): p. 2005.
81. Askarzadeh, A., *A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm.* Computers & Structures, 2016. **169**: p. 1-12.

82. He, Q. and L. Wang, *An effective co-evolutionary particle swarm optimization for constrained engineering design problems.* Engineering applications of artificial intelligence, 2007. **20**(1): p. 89-99.
83. Mezura-Montes, E. and C.A.C. Coello, *An empirical study about the usefulness of evolution strategies to solve constrained optimization problems.* International Journal of General Systems, 2008. **37**(4): p. 443-473.
84. Coello, C.A.C., *Use of a self-adaptive penalty approach for engineering optimization problems.* Computers in Industry, 2000. **41**(2): p. 113-127.
85. Akay, B. and D. Karaboga, *Artificial bee colony algorithm for large-scale problems and engineering design optimization.* Journal of intelligent manufacturing, 2012. **23**(4): p. 1001-1014.
86. Kaveh, A. and M. Khayatazad, *A new meta-heuristic method: ray optimization.* Computers & structures, 2012. **112**: p. 283-294.
87. Huang, F.-z., L. Wang, and Q. He, *An effective co-evolutionary differential evolution for constrained optimization.* Applied Mathematics and Computation, 2007. **186**(1): p. 340-356.
88. Lee, K.S. and Z.W. Geem, *A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice.* Computer methods in applied mechanics and engineering, 2005. **194**(36-38): p. 3902-3933.
89. Arora, J.S., *Introduction to optimum design*. 2004: Elsevier.
90. Kaveh, A. and S. Talatahari, *An improved ant colony optimization for constrained engineering design problems.* Engineering Computations, 2010.
91. Mirjalili, S., S.M. Mirjalili, and A. Hatamlou, *Multi-verse optimizer: a nature-inspired algorithm for global optimization.* Neural Computing and Applications, 2016. **27**(2): p. 495-513.
92. Coello, C.A.C. and E.M. Montes, *Constraint-handling in genetic algorithms through the use of dominance-based tournament selection.* Advanced Engineering Informatics, 2002. **16**(3): p. 193-203.
93. Mezura-Montes, E. and C.A.C. Coello. *Useful infeasible solutions in engineering optimization with evolutionary algorithms*. in *Mexican international conference on artificial intelligence*. 2005. Springer.
94. Czerniak, J.M., H. Zarzycki, and D. Ewald, *AAO as a new strategy in modeling and simulation of constructional problems optimization.* Simulation Modelling Practice and Theory, 2017. **76**: p. 22-33.
95. Baykasoğlu, A. and Ş. Akpinar, *Weighted Superposition Attraction (WSA): A swarm intelligence algorithm for optimization problems – Part 1: Unconstrained optimization.* Applied Soft Computing, 2017. **56**: p. 520-540.
96. Mirjalili, S., *Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm.* Knowledge-Based Systems, 2015. **89**: p. 228-249.
97. Abualigah, L., et al., *The Arithmetic Optimization Algorithm.* Computer Methods in Applied Mechanics and Engineering, 2021. **376**: p. 113609.
98. Liu, H., Z. Cai, and Y. Wang, *Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization.* Applied Soft Computing, 2010. **10**(2): p. 629-640.
99. Baykasoğlu, A. and F.B. Ozsoydan, *Adaptive firefly algorithm with chaos for mechanical design optimization problems.* Applied Soft Computing, 2015. **36**: p. 152-164.
100. Adeli, H. and O. Kamal, *Efficient optimization of space trusses.* Computers & Structures, 1986. **24**(3): p. 501-511.
101. Kaveh, A. and S. Talatahari, *Size optimization of space trusses using Big Bang–Big Crunch algorithm.* Computers & Structures, 2009. **87**(17): p. 1129-1140.
102. Cao, G., *Optimized design of framed structures using a genetic algorithm*. 1996: The University of Memphis.

103. Camp, C.V. and B.J. Bichon, *Design of space trusses using ant colony optimization.* Journal of structural engineering, 2004. **130**(5): p. 741-751.
104. Perez, R.E. and K. Behdinan, *Particle swarm approach for structural design optimization.* Computers & Structures, 2007. **85**(19): p. 1579-1588.
105. Camp, C.V., *Design of space trusses using Big Bang–Big Crunch optimization.* Journal of Structural Engineering, 2007. **133**(7): p. 999-1008.