IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 3, 2019, accepted May 15, 2019, date of publication May 23, 2019, date of current version June 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2918753

Supply-Demand-Based Optimization: A Novel Economics-Inspired Algorithm for Global Optimization

WEIGUO ZHAO[1,2], LIYING WANG[1], AND ZHENXING ZHANG[2]
[1]School of Water Conservancy and Hydropower, Hebei University of Engineering, Handan 056021, China
[2]Illinois State Water Survey, Prairie Research Institute, University of Illinois at Urbana-Champaign, Champaign, IL 61820, USA

Corresponding author: Liying Wang (wangliying@hebeu.edu.cn)

This work was supported in part by the Natural Science Foundation of Hebei Province of China under Grant E2018402092 and Grant F2017402142, and in part by the Scientific Research Key Project of University of Hebei Province of China under Grant ZD2017017.

ABSTRACT A novel metaheuristic optimization algorithm, named supply-demand-based optimization (SDO), is presented in this paper. SDO is a swarm-based optimizer motivated by the supply-demand mechanism in economics. This algorithm mimics both the demand relation of consumers and supply relation of producers. The proposed algorithm is compared with other state-of-the-art counterparts on 29 benchmark test functions and six engineering optimization problems. The results on the unconstrained test functions prove that SDO is able to provide very promising results in terms of exploration, exploitation, local optima avoidance, and convergence rate. The results on the constrained engineering problems suggest that SDO is considerately competitive in terms of computational expense, convergence rate, and solution accuracy. The codes are available at https://www.mathworks.com/matlabcentral/fileexchange/71764-supply-demand-based-optimization.

INDEX TERMS Supply-demand-based optimization, global optimization, engineering design, constrained problems, optimization algorithm, particle swarm optimization, swarm intelligence.

I. INTRODUCTION

Optimization is to find optimal solutions with the most cost effective form a given solution space under the given constraints, by either maximizing or minimizing its objective function. Over the last few decades, the substantial increase of optimization problems from different fields makes optimization techniques become a major research area for the scientific community. The conventional optimization techniques, including steepest descent method [1] and Newton's method [2], are developed. However, owing to their inherent drawbacks of the requirement of derivative and local search stagnation, they tend to become less powerful when solving complex nonlinear functions and constraints with multiple peaks [3], [4]. So, some stochastic optimization methods have emerged and have been widely applied in a wide range of fields [5]–[9]. Stochastic optimization methods refer to that random variables are used to generate to random objective functions and constraint values. Their outstanding merit is randomness, which is able to help these methods easily bypass local optima and explore the entire variable space. Another merit of stochastic optimization methods is that they treat considered problems as a black box, which means that one only needs to lay emphasis on the input and output of objective functions instead of their gradient information. Additionally, stochastic optimization methods are highly versatile and flexible, implying their extendibility and practicability to different types of optimization problems.

General speaking, there are two different classification in stochastic optimization methods. One classification is based on the number of their random solutions over the course of iterations, they are divided into single-solution-based and swarm-based algorithms [10]. For the former, stochastic algorithms randomly generate a single solution and update it by introducing minor change in an iterative process. Simulate annealing (SA) [11] belongs to this kind of method. For the latter, stochastic algorithms randomly initialize a set of solutions rather than one solution and then manipulate them at each stage. The major difference between single-solution-based and swarm-based algorithms is that the later

The associate editor coordinating the review of this manuscript and approving it for publication was Chee Keong Kwoh.

is able to share the solution information among the swarm and improve the quality of solutions by introducing major change in subsequent iterations. This assists swarm-based algorithms explore the search space and avoid the local optima. The other classification for stochastic optimization methods is based on their inspirations and motivations, and they are classified into evolutionary-based (EB) [12], swarm-based (SB) [13], physics-based (PB) [14], and human-based (HB) [15] algorithms.

EB is generic population-based metaheuristic that simulates biological evolution i.e., selection, crossover, mutation, recombination, and chemotaxis [16]. The most well-known EB is genetic algorithm (GA) [17]. GA randomly initializes a group of agents and computes their fitness values. The selection, crossover, and mutation operators are used to improve each agent of a population. The agent with the best fitness value is stored and employed to generate a new population over the course of iterations. Another popular EB is differential evolution (DE) [18]. Similar to GA, DE randomly generates a population that iteratively undergoes mutation and recombination as well as becomes subject to a selection operator. Both GA and DE tend to evolve the population by mimicking the survival of the fittest in nature, therefore, they are always able to obtain high-quality solutions and avoid local optima. Apart from these, some other well-known EB techniques are evolution strategy (ES) [19], genetic programming (GP) [20], selfish gene algorithm (SGA) [21], shuffled frog leaping algorithm (SFLA) [22], biogeography-based optimization (BBO) [23], and fruit fly optimization algorithm (FOA) [24].

PB usually manipulates and improves a population by physical lows in nature, including gravitational force, electromagnetic force, energy conservation, and momentum conservation. The general mechanism of PB is different from the other techniques since agents communicate and exchange information according to physical laws. SA [11] and gravitational search algorithm (GSA) [25] are viewed as two representatives in PB. SA is originated from a physical process in which a material is heated and then its temperature is slowly lowered to minimize the system energy. During this process, SA searches solutions depending on a probability with a scale proportional to the temperature. Namely, SA not only retains all the new solutions improving their fitness values, but also retains the new solutions lowering their fitness values with a certain probability. By retaining these solutions lowering their fitness values, SA can avoid the local optima and globally explore for more possible solutions. However, this algorithm probably suffers from more computational expense especially when the objective functions are very complex or high-dimensional in nature. GSA is based on the Newton's famous law of gravity and the law of motion. Depending to the law of motion, each agent attracts towards each other. A lighter agent indicates a worse solution and a heavier agent indicates a better solution. The higher agents always attract towards the heavier ones by the gravity force, causing a global movement. This particular movement mechanism gives GSA an advantage over some of the other algorithms in terms of global search, local optima avoidance, and solution accuracy. However, this algorithm tends to suffer from slow searching speed in the later iterations and a complex operator [26]. Some other emerging PBs include big bang-big crunch algorithm (BB-BC) [27], colliding bodies optimization (CBO) [28], charged system search (CSS) [29], wind driven optimization (WDO) [30], central force optimization (CFO) [31], water evaporation optimization (WEO) [32], galaxy-based search algorithm (GBSA) [33], electromagnetic field optimization (EFO) [34], atom search optimization (ASO) [35], artificial physics optimization (APO) [36], and thermal exchange optimization(TEO) [37].

Different from EBs and PBs, SBs always stimulate collective behaviors of social creatures to offer the better solutions to investigated problems. The inspiration of SBs generally comes from natural colonies, flocks, and herds. Two of the most classic SBs are particle swarm optimization (PSO) and ant colony optimization (ACO).

In PSO, a swarm of particles perform a search for the optimal solution and each of them iteratively updates its solution using its personal and social information in the population. The main merit of PSO is the faster convergence rate and less computational cost. The major drawback of PSO is that it easily suffers from convergence prematurely and traps into the local optima especially with complex multimodal functions [38]. ACO is motivated from the social behavior of ants when foraging. Each ant is subject to finding the shortest route between nest and the source food via pheromone trails. ACO is able to perform searching among a population parallel, so it is very good at solving the salesman problem. However, the major disadvantage behind this algorithm is that it has uncertain convergence time and difficult theoretical analysis. The other popular SBs are hunting search (HS) algorithm [39], tree-seed algorithm (TSA) [40], whale optimization algorithm (WOA) [41], cuckoo search (CS) [42], crow search algorithm (CSA) [43], dolphin echolocation (DE) algorithm [44], firefly algorithm (FA) [45], virus colony search (VCS) [46], bird mating algorithm (BMO) [47], emperor penguin optimizer (EPO) [48], and krill herd algorithm (KH) [49].

HBs are a new developed category in intelligence computing recently, they mathematically stimulate social activities and ideology in humans to find near optimal solutions. Society and civilization algorithm (SCA) [50] is a typical representative of HBs. SCA imitates the intra and social interactions within a formal society and the civilization model. A society corresponds to a set of mutually interacting individuals and a civilization is a set of all such societies. All individuals in each society interact with each other and make improvements under the guidance of a leader belonging to the same society. Meanwhile, each leader interacts with leaders of other societies to migrate to a developed society. This leader migration mechanism helps SCA globally search promising regions in the variable space. Additionally, SCA is advantageous in terms to dealing with constrained optimization problems owing to the leader identification

mechanism. Some of other HBs include league championship algorithm (LCA) [51], social group optimization (SGO) [52], social emotional optimization algorithm (SOA) [53], socio evolution and learning optimization algorithm (SELO) [54], ideology algorithm (IA) [55], and cultural evolution algorithm (CEA) [56].

SBs have some advantages over the other three categories. SBs are more able to control convergence and need less computational cost owing to the less number of operators; SBs are algorithmically simpler with less control parameters and have a better convergence rate; SBs are able to restore and share part or all of historical information about the population more effectively, guiding all the individuals to perform a global search.

For all the heuristic algorithms, it is very important to effectively perform both exploration and exploitation. Exploration refers to probing a much larger region of the search space with the hope of finding other promising solutions whereas exploitation refers to probing a promising region of the search space with the hope of improving a promising solution [57], [58]. But, it is challenging to properly balance between exploration and exploitation in the development of heuristic algorithms because of their stochastic nature. Therefore, this is one fact that motivates us to develop an effective optimizer to tackle real-world problems.

One might be asking why new optimization methods are still raised despite of so many existing algorithms. The answer can be found in the No Free Lunch Theorem of Optimization [59], which states that an optimization algorithm that can well solve a certain optimization problem does not guarantee to successfully deal with the other different optimization problems. Therefore, developing new and effective swarm-inspired optimizers to tackle specific real-world problems also motivates this study.

Inspired by the supply-demand mechanism in economics, this study proposes a new metaheuristic algorithm called supply-demand-based optimization (SDO). SDO mathematically simulates the demand relation of consumers and the supply relation of producers. The performance of SDO is tested using both unconstrained benchmark functions and constrained engineering problems.

The paper is organized as follows. Section II presents the inspiration and SDO algorithm in detail. The experimental results on 29 mathematical benchmark functions are analyzed in Section III. In Section IV, the effectiveness of SDO in solving 6 engineering problems is investigated. Section V concludes the work and suggests several directions for future work.

II. SUPPLY-DEMAND-BASED OPTIMIZATION (SDO)

In this section, the inspiration of SDO is firstly introduced. Then its mathematical model is provided in detail.

A. INSPIRATION

According to economic theory, the commodity price and commodity quantity in a market might firstly be subject to periodic fluctuations and then gradually be stable onto their respective equilibrium points [60]. This process generally depends on both the supply relation of producers and demand relation of consumers.

The supply-demand mechanism is an economic theory of price determination in a market economy. According to this theory, a commodity quantity in the next time q_{t+1} depending on its current price p_t in a market is determined by the supply relation of producers, that is $q_{t+1} = f(p_t)$, here f is a linear supply function. When the current commodity price increases in a market, the commodity quantity will increase in supply at the next time, so f is an increasing function. The commodity price p_{t+1} in the next time depending on its quantity q_{t+1} in the same time is determined by the demand relation of consumers, this is $p_{t+1} = g(q_{t+1})$, here g is a linear demand function. When the commodity quantity increases, its price in a market will decrease, so g is a decreasing function. With the oscillations, these two functions finally intersect at a point $P(x_0, y_0)$, which is called the equilibrium point. x_0 and y_0 are the equilibrium price and equilibrium quantity of a commodity, respectively.

The supply function f can be expressed as [61]

$$q_{t+1} - q_0 = a(p_t - p_0) \tag{1}$$

The demand function g can be expressed as [61]

$$p_{t+1} - p_0 = -b(q_{t+1} - q_0) \tag{2}$$

where t is the time, a and b are the linear coefficients.

Generally, the supply-demand mechanism has two modes. One is the stability mode as shown in Fig. 1(A). When $|ab|<1$, the supply function f is steeper than the demand function g, then the oscillations reduce in magnitude with each time, so the curve of the commodity price and quantity with respects to time tends to spiral inwards. After a certain time of iterations, the price and quantity converge to the equilibrium point (x_0, y_0). The other is the instability mode as shown in Fig. 1(B). When $|ab|>1$, the demand function g is steeper than the supply function f, then the oscillations increase in magnitude with each time, a curve of the price and quantity with respects to time tends to spiral onwards. Therefore, the price and quantity diverge from the equilibrium point (x_0, y_0) more and more as time passes. Because these spiral curves often look like a cobweb, this demand-supply mechanism is named the cobweb model in economics [60]. The cobweb model is a very famous economic theory in which the price fluctuation results in fluctuation in supply, thus causing a periodic rising or falling in price. Therefore, it is extensively used for studying the price fluctuation of various products in certain types of markets.

B. SUPPLY-DEMAND-BASED OPTIMIZATION (SDO)

According to the supply-demand mechanism, the stability mode can encourage both the commodity price and quantity to exploit the neighborhood of the equilibrium point and this exploiting process is oscillating in magnitude over time. While the instability mode tends to force both the commodity

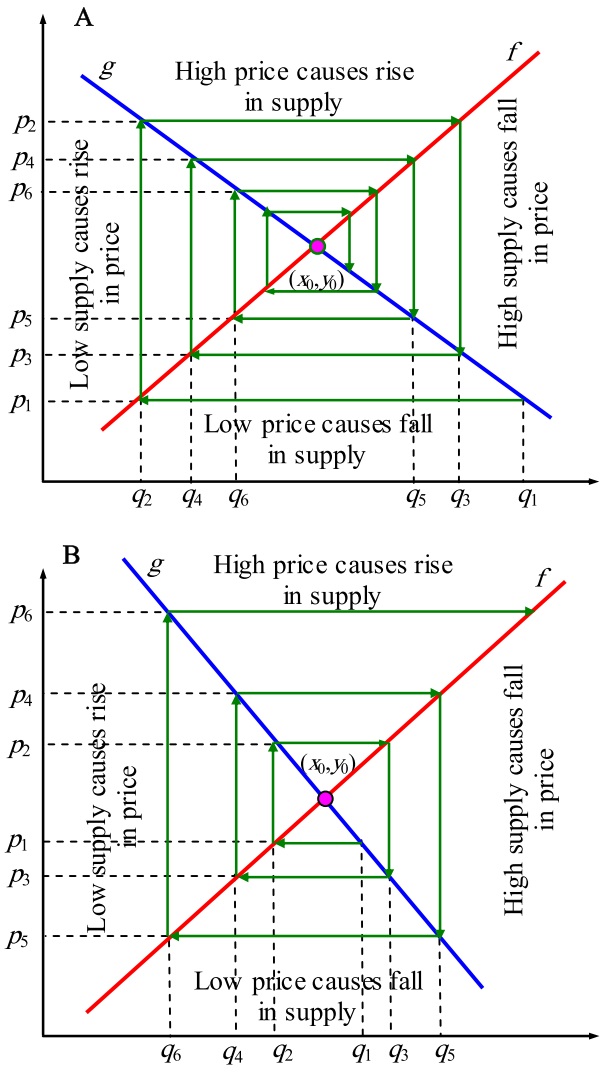

FIGURE 1. Two modes of supply-demand mechanism: (A) stability mode and (B) instability mode.

price and quantity to gradually explore new regions far away from the equilibrium point. The stability mode gradually decreasing oscillations in the supply-demand mechanism can easily be introduced to SDO as exploitation to perform a local search in a promising region. Similarly, the instability mode can be borrowed to SDO as exploration to globally perform a search in the search space. Fig. 2 shows the introduction from the supply-demand mechanism in economic theory to the proposed SDO algorithm.

For SDO algorithm, suppose there are n markets, each of which has d kinds of different commodities, and each kind of commodity has a certain quantity and price. The d commodity prices of a market represent a candidate solution as d variables to the optimization problem, and the d commodity quantities of a market are reviewed as a possible candidate solution. If this possible candidate solution is better than the candidate solution, the candidate solution is replaced with the possible one. Because the proposed method is a swarm-based

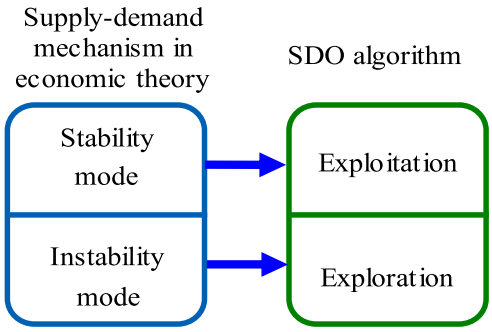

FIGURE 2. Introduction form supply-demand mechanism to SDO.

algorithm, the commodity price and the commodity quantity are given in two matrixes, respectively.

The commodity price matrix of markets is given as

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^d \\ x_2^1 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^d \end{bmatrix} \tag{3}$$

where d is the number of commodity prices (variables) in each market, n is the number of markets (candidate solutions), x_i^j $(i = 1,\ldots, n; j = 1,\ldots, d)$ is the jth commodity price in the ith market, and $\boldsymbol{x_i}$ $(i = 1,\ldots, n)$ is the ith commodity price vector corresponding to a candidate solution.

The commodity quantity matrix of markets is given as

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} y_1^1 & y_1^2 & \cdots & y_1^d \\ y_2^1 & x_2^2 & \cdots & y_2^d \\ \vdots & \vdots & \vdots & \vdots \\ y_n^1 & y_n^2 & \cdots & y_n^d \end{bmatrix} \tag{4}$$

where d is the number of commodity quantities in each market, y_i^j $(i = 1,\ldots, n; j = 1,\ldots, d)$ indicates the jth commodity quantity in the ith market, and $\mathbf{y_i}$ $(i = 1,\ldots, n)$ is the ith commodity quantity vector corresponding to a possible candidate solution.

The commodity price vector and quantity vector of each market are evaluated using the fitness function, respectively. The fitness value calculated by each price vector is stored as the return value of the fitness function for each market. Yet the fitness value calculated by each commodity quantity vector is also stored and only used for the solution replacement of a market according to its quality. For all the markets, an array for storing the fitness values of commodity price vectors in n markets is given

$$Fx = [Fx_1 \quad Fx_2 \quad \cdots \quad Fx_n]^T \tag{5}$$

Another array for storing the function values of commodity quantity vectors in n markets is given

$$Fy = [Fy_1 \quad Fy_2 \quad \cdots \quad Fy_n]^T \tag{6}$$

where T signifies the transpose of the array.

Although the commodity price vector and quantity vector in each market are viewed as both solutions, the ways that

they are treated and updated are different during the iterations. Meanwhile, the concepts of both the stability and instability modes in the supply-demand theory are utilized to perform exploration and exploitation in SDO, respectively.

The commodity equilibrium price $\boldsymbol{x_0}$ and equilibrium quantity $\boldsymbol{y_0}$ are two important components in SDO, yet they are not known a priori during the iterations, therefore, they need to be firstly designated, respectively. From the above section, it should be noted that the updating for both the demand function and supply function of a commodity is respectively based on both a fixed equilibrium price and a fixed equilibrium quantity from beginning to end. However, this updating will make the algorithm trap into the local extrema quickly and result in premature convergence. To prevent them, the equilibrium price of each commodity should be variable at each iteration, likewise with the equilibrium quantity of each commodity. Specifically, for each iteration, each market chooses a commodity quantity vector from the quantity array as its quantity equilibrium vector by means of its probability, and the better the fitness value of the quantity vector in a market, the higher the probability of that the quantity vector chosen is. Meanwhile, each market also chooses either a price vector from the price array according to its probability or the average of commodity price vectors of all the markets as the equilibrium price vector to improve exploration. The equilibrium quantity vector $\boldsymbol{y_0}$ is represented as follows

$$N_i = \left| Fy_i - \frac{1}{n}\sum_{i=1}^{n} Fy_i \right| \tag{7}$$

$$Q = \frac{N}{\sum_{i=1}^{n} N_i} \tag{8}$$

$$y_0 = y_k, \quad k = \text{RouletteWheelSelection}(Q) \tag{9}$$

The equilibrium price vector $\boldsymbol{x_0}$ is represented as follows

$$M_i = \left| Fx_i - \frac{1}{n}\sum_{i=1}^{n} Fx_i \right| \tag{10}$$

$$P = \frac{M}{\sum_{i=1}^{n} M_i} \tag{11}$$

$$x_0 = \begin{cases} r_1 \cdot \dfrac{\sum_{i=1}^{n} x_i}{n} & \text{if } rand < 0.5 \\ x_k, \ k = \text{RouletteWheelSelection}(P) & \text{if } rand \geq 0.5 \end{cases} \tag{12}$$

where r_1 is a random number in [0, 1]. According to (12), SDO assumes that there is a probability of 50% to choose either the average of commodity price vector or a price vector depending to its probability in the price array as the equilibrium price vector. With the definitions of both the equilibrium price vector $\boldsymbol{x_0}$ and equilibrium quantity vector $\boldsymbol{y_0}$, the supply function and demand function are proposed as follows, respectively

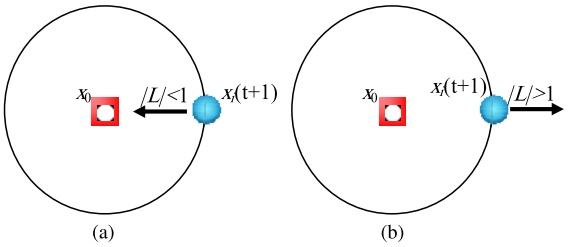

FIGURE 3. (a) Stability mode and (b) instability mode of SDO algorithm.

$$y_i(t+1) = y_0 + \alpha \cdot (x_i(t) - x_0) \tag{13}$$

$$x_i(t+1) = x_0 - \beta \cdot (y_i(t+1) - y_0) \tag{14}$$

where $\boldsymbol{x_i}(t)$ is the ith commodity price vector at the time t, $\boldsymbol{y_i}(t)$ is the ith commodity quantity vector at the time t, and α and β are the supply weight and demand weight, respectively.

From (13), the commodity quantity vector in each market can be updated according to both the equilibrium quantity vector and equilibrium price vector. Plugging (13) into (14) the demand equation can be rewritten as

$$x_i(t+1) = x_0 - \alpha\beta \cdot (x_i(t) - x_0) \tag{15}$$

It can be found that from this equation, the commodity price vector is updated with respect to its current price vector according to the equilibrium price vector actually. Different commodity price vectors can be obtained by adjusting the values of weights α and β. To perform exploration and exploitation in SDO, both the supply weight α and demand weight β need to be appropriately presented. These two weights can be formulated as

$$\alpha = \frac{2 \cdot (T - t + 1)}{T} \cdot \sin(2\pi r) \tag{16}$$

$$\beta = 2 \cdot \cos(2\pi r) \tag{17}$$

where T is the maximum number of iterations and r is a random number or vector in [0, 1]. Let the variable L equal the product of weights α and β, we can get

$$L = \alpha\beta = \frac{4 \cdot (T - t + 1)}{T} \cdot \sin(2\pi r)\cos(2\pi r) \tag{18}$$

where, $|L| < 1$ corresponds to the stability mode as depicted in Fig. 3(A), different commodity price vectors around the equilibrium price $\boldsymbol{x_0}$ can be obtained with respect to the current price vector by adjusting the weights α and β, and these price vectors are possible to be randomly changed between the current price vector and the equilibrium price vector by the random number or vector r. This is to say, the algorithm allows each market to update its all commodity prices in the neighborhood of the current commodity prices and mimic the stability mode in Fig. 1(A). This mechanism emphasizes exploitation and encourages SDO algorithm to search locally. As depicted in Fig. 3(B), $|L| > 1$ corresponds to the instability mode that allows the commodity price vector in any market to move far away from the equilibrium price vector,

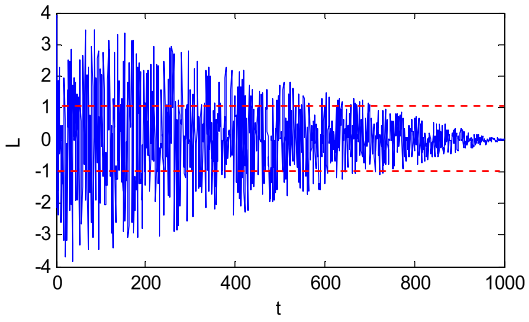

FIGURE 4. Values of variable *L* over iterations.

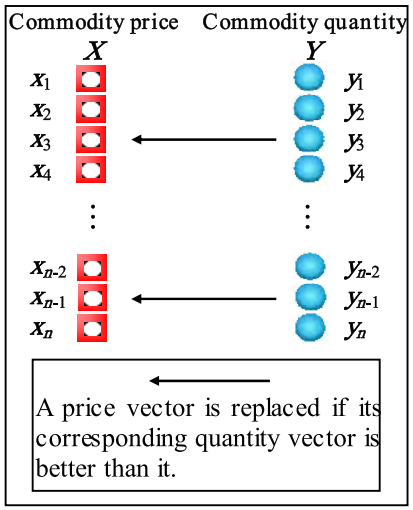

FIGURE 5. Solution replacement mechanism.

Initialize the market population and weights: the commodity price vectors x_i and commodity quantity vectors y_i are randomly initialized, calculate their fitness values Fx_i and Fy_i, replace x_i by y_i if Fy_i is better Fx_i, and x_{best}= the best solution found so far. The weights α and β are set.

While the stop criterion is not satisfied do

For each market (i=1,...,n)

Determine the equilibrium quantity y_0 by equations 7-9 and the equilibrium price x_0 by equations 10-12, respectively.

Update the commodity quantity vector y_i by equation 13.

Update the commodity price vector x_i by equation 14.

Calculate their fitness values Fx_i and Fy_i.

If Fy_i is better than Fx_i,

replace x_i by y_i,

End If.

End For.

Update the best solution found so far x_{best}.

End While.

Return the best solution found so far x_{best}.

FIGURE 6. Pseudo code of SDO algorithm.

this forces each market to search new promising regions in the search space and simulate the instability mode in Fig. 1(B). This mechanism concentrates on exploration and forces SDO algorithm to search globally.

The values of the variable L over iterations are depicted in Fig. 4 where the maximum number of iterations T is set as 1000. As observed in this figure, in early iterations, the values of L are greater than 1 or less than -1 with the high probability. As the iterations increase, this high probability begins to decrease, and there is an increasing probability that the function values of L are in [-1, 1]. In later iterations, the values of L are in [-1, 1] with the increasingly high probability. Obviously, SDO gets high exploration in early stage of iterations and smoothly switches to high exploitation in later stage of iterations.

After updating both the commodity price and commodity quantity vectors in each iteration, they need to be evaluated by the fitness function and the results are stored in their respective arrays. If the fitness value of the ith commodity quantity vector is better than that of the i th commodity price vector, the ith commodity price vector will be replaced with this quantity vector as a candidate solution. Fig. 5 shows this solution replacement mechanism.

SDO starts the optimization by creating a set of markets randomly. At each iteration, each commodity quantity of a market is updated with respect to both the equilibrium price and equilibrium quantity, and then each commodity price of a market is updated with respect to the equilibrium price. The equilibrium price vector can be randomly switched between a chosen commodity price vector form the price array according to its probability in the price array and the average of commodity price vectors. The equilibrium quantity vector is chosen from the quantity array according to its probability. These updates for commodity prices and quantities are achieved by adjusting the values of weights both α and β. The values of the variable L linearly decrease with random fluctuation in order to perform either exploration or exploitation. The commodity prices of a market is inclined to diverge from the equilibrium price when $|L| > 1$ and converge towards the equilibrium price when $|L| < 1$. Then the updated price vectors and quantity vectors are evaluated by the fitness function. For each market, if the fitness value of its commodity quantities is better than that of its commodity prices, its commodity prices will be replaced with its commodity quantities as a candidate solution. Eventually, the best commodity price vector of a market as the best solution found so far is returned when the stop criterion is satisfied. Fig. 6 gives the pseudo code of

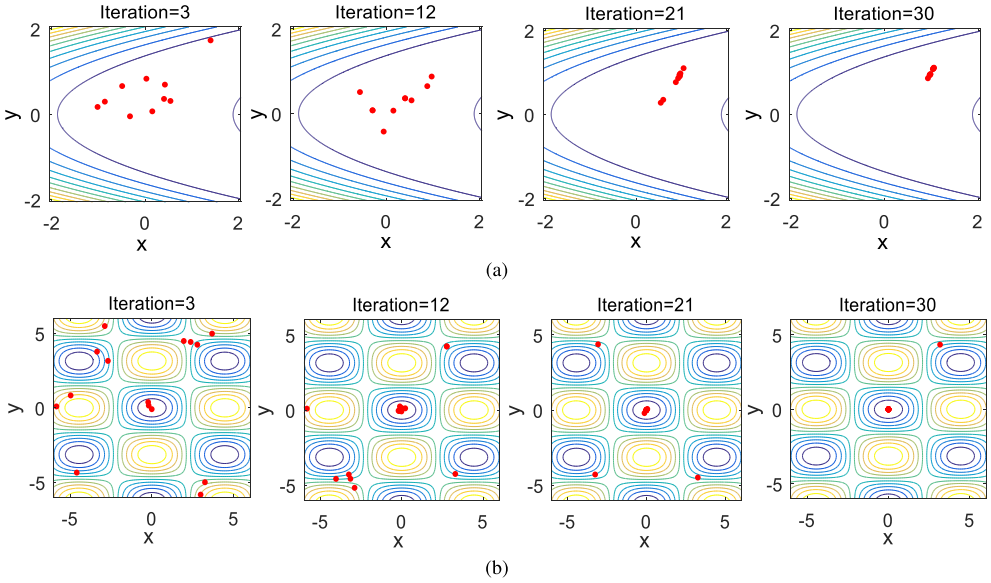

FIGURE 7. Swarm search of SDO in a two-dimensional space. (a) Unimodel function (Rosenbrock). (b) Multimodel function (Griewank).

SDO algorithm and Fig. 7 shows an effective swarm search of SDO for both the unimodal and multimodal functions in a 2-D space.

With the above formulation of SDO algorithm and the observation of its optimization performance, the following remarks are made.

(1) The SDO updates two different equations, one is the supply equation updating the commodity quantity, and the other is the demand equation updating the commodity price.
(2) In each iteration, both the commodity equilibrium price and equilibrium quantity need to be determined. The equilibrium price vector is randomly chosen either the average of commodity price vectors or a commodity price vector according to its probability in the price array. The equilibrium quantity vector is chosen from the quantity array according to its probability.
(3) The commodity quantity vector is updated based on the commodity price vector with respect to both the equilibrium price vector and equilibrium quantity vector by adjusting the supply weight. The commodity price vector is updated based on the commodity price vector with respect to both the equilibrium price vector and equilibrium quantity vector by adjusting the demand weight.
(4) SDO tends to force the commodity prices of a market to diverge from the equilibrium quantities when $|L| > 1$ and converge towards the equilibrium prices when $|L| < 1$.
(5) The values of variable L assist SDO to smoothly transit between exploration and exploitation.
(6) With the decrease of variable L, in early iterations, SDO emphasizes exploration, yet in later iterations, SDO focus on exploitation.
(7) In each iteration, when the fitness value of a commodity price vector is worse than that of its corresponding commodity quantity vector, this commodity price vector will be replaced.
(8) SDO is very simple to implement and require few parameters to be adjusted.

The time complexity of SDO depends on the number of variables, number of markets, maximum number of iterations, roulette wheel selection, replacement, and updates of commodity price and quantity in each iteration. We utilize the roulette wheel selection in both the demand function and supply function, which is of $O(n^2)$ in the worst case, and the replacement is performed for each market over iterations and is of $O(nd)$ in the worst case. Therefore, the overall time complexity of this algorithm is given as

$$\begin{aligned} &O(SDO) \\ &= O(T(O(\text{Roulettewheel in choosing equilibrium price}) \\ &\quad + O(\text{Roulettewheel in choosing equilibrium quantity}) \\ &\quad + O(\text{Price update})+\text{O(Quantity update)} \\ &\quad + O(\text{Replacement}))) \end{aligned} \tag{19}$$

$$\begin{aligned} &O(SDO) \\ &= O(T(\frac{1}{2}n^2 + n^2 + nd + nd + nd)) \\ &= O(Tn^2) + O(Tnd) \end{aligned} \tag{20}$$

where n is the number of markets, T is the maximum number of iterations, and d is the number of commodities in each market.

III. EXPERIMENTAL RESULTS AND DISCUSSION

A. BENCHMARK FUNCTIONS

A set of comprehensive benchmark functions with known solutions are used to test the performance of algorithms.

TABLE 1. Result comparisons of algorithms for unimodal functions.

Function	Index	SDO	PSO	GA	DE	CS	GSA	ABC
$f_1(x)$	Mean	2.526E-162	0.000215	0.009247	3.636E-14	0.009675	2.195E-17	0.002363
	Std	7.555E-162	0.000225	0.003778	6.064E-14	0.004518	6.379E-18	0.001525
$f_2(x)$	Mean	4.714E-68	0.000296	0.021264	4.385E-08	1.403794	2.283E-08	2.317E-04
	Std	5.156E-68	0.000231	0.005609	2.530E-08	0.560948	3.490E-09	1.530E-04
$f_3(x)$	Mean	7.192E-130	2.844E+03	1.053E+03	5.693864	4.725E+02	2.216E+02	9.562E+03
	Std	3.189E-130	1.343E+03	3.489E+02	3.909791	1.096E+02	70.667942	1.750E+03
$f_4(x)$	Mean	7.152E-77	17.363071	1.046239	9.171683	3.247527	3.451E-09	24.533250
	Std	2.495E-77	3.623843	0.289725	3.998583	0.854660	7.445E-10	2.284116
$f_5(x)$	Mean	25.501782	94.730855	98.563643	3.000E+01	38.611217	26.694542	5.470E+02
	Std	0.302722	78.960299	58.445847	1.765E+01	10.287210	2.671336	2.099E+02
$f_6(x)$	Mean	0	0.133333	0.000000	0.133333	0.000000	0.000000	0.000000
	Std	0	0.434172	0.000000	0.434172	0.000000	0.000000	0.000000
$f_7(x)$	Mean	1.435E-04	0.056438	0.044730	0.214589	0.030850	0.019135	0.095282
	Std	8.216E-05	0.020323	0.013553	0.072382	0.007932	0.006870	0.023852

We adopt 29 mathematical functions in literature [62]–[64] for comparison. These functions are categorized into four types: unimodal, multimodal, fixed-dimensional, and composite. The unimodal functions (f_1-f_7) have only one global optimum and no local optima, hence the convergence rate and exploitation of algorithms can be examined. Yet multimodal functions (f_8-f_{13}) have a considerable number of local optima and low-dimensional functions (f_{14}-f_{23}) have few local optima, so both are used to benchmark exploration and local extrema avoidance of algorithms. Finally, the composite functions (f_{24}-f_{29}) are complex multimodal functions combining shifted, rotated, expanded and biased functions, and their detailed description in the CEC 2005 special session is available in [65]. These composite functions are more challenging than unimodal and multimodal functions because they can shift the global optima from the specific position to random position before each iteration and occasionally relocate the global optima on the boundary of search space. Therefore, they are especially adapted to test the balance between exploration and exploitation of algorithms. All the employed benchmark functions are described in Appendix A.

B. EXPERIMENT SETUP AND COMPARATIVE ALGORITHMS

In order to test the performance of the proposed SDO algorithm, some classic and recently proposed stochastic algorithms are employed for comparison: GA, PSO, DE, CS, ABC and GSA. Although numerous variants of algorithms have been developed, the comparisons of standard versions can be used to interpret the results of larger groups. In this test, the population size is set as 50 for all the tested algorithms. In addition, every algorithm runs 30 times for each function and implements 50,000 function evaluations (FEs) in each run. The results are based on the average performance of these runs. Two performance evaluation indexes are used to quantitatively compare all the algorithms: the average and standard deviation of best-so-far solutions. The parameter values of each algorithm are presented as follows.

PSO: Inertia coefficient linearly reduces from 0.9 to 0.2, acceleration coefficients $c_1 = 2$ and $c_2 = 2$.

GA: Decreasing coefficient $\gamma = 20$, mutation rate $p_m = 0.2$, crossover rate $p_c = 0.8$ and crossover adopts roulette wheel method.

DE: Mutation factor $F = 0.5$ and crossover rate $C = 0.5$.

CS: Mutation probability $p_a = 0.25$.

GSA: Initial gravitational constant $G_0 = 100$ and decreasing coefficient $a = 20$.

ABC: Limit parameter $= n \cdot d$.

C. ANALYSIS OF EXPLOITATION CAPABILITY

The optimization results offered by the employed optimizers on unimodal functions are described in Table 1, in which 'Mean' indicates the mean of best-so-far solutions and 'Std' indicates the standard deviation of best-so-far solutions. As shown in Table 1, SDO provides highly competitive solutions in terms of the 'Mean' and 'Std' indexes than all other competitors. These results discover that SDO is more effective than its competitors in finding the best optimal solutions, demonstrating its superior search ability in terms of exploitation. This merit results from the exploitation mechanism in SDO previously discussed.

D. ANALYSIS OF EXPLORATION CAPABILITY

Differing from unimodal functions, both multimodal and fixed-dimensional functions are able to used to evaluate exploration of algorithms. The optimization results offered by different optimizers on multimodal and fixed-dimensional functions are reported in Tables 2 and 3, respectively. From these tables, SDO can obtain the best results of all the investigated algorithms for all the functions but functions f_{12}, f_{13} and f_{20}. However, SDO ranks only second to GA on both functions f_{12} and f_{13}, as well as performs better than PSO,

TABLE 2. Result comparisons of algorithms for multimodal functions.

Function	Index	SDO	PSO	GA	DE	CS	GSA	ABC
$f_8(x)$	Mean	-8.740E+03	-5.139E+03	-6.820E+03	-5.310E+03	-8.692E+03	-2.639E+03	-5.124E+03
	Std	6.856E+02	5.777E+02	5.769E+02	6.617E+02	2.352E+02	4.351E+02	4.607E+02
$f_9(x)$	Mean	0	30.875576	12.400190	1.650E+02	83.232445	15.853010	1.578E+02
	Std	0	8.883905	2.874732	17.511476	13.099442	3.848434	21.302371
$f_{10}(x)$	Mean	8.882E-16	0.007599	0.020177	5.413E-08	4.152485	3.443E-09	0.053290
	Std	0	0.008229	0.004771	2.618E-08	1.486545	5.587E-10	0.040472
$f_{11}(x)$	Mean	0	0.014768	0.021579	0.002054	0.096258	4.264E+00	0.163104
	Std	0	0.013605	0.010212	0.003888	0.041825	1.587503	0.115192
$f_{12}(x)$	Mean	2.568E-04	0.375773	2.977E-05	0.006911	1.100793	0.033992	15.014029
	Std	8.839E-06	0.725728	3.014E-05	0.026302	0.317227	0.053609	4.671204
$f_{13}(x)$	Mean	8.455E-03	0.190921	6.310E-04	0.053060	0.134335	2.041E-18	36.258923
	Std	5.534E-03	0.387922	4.234E-04	0.288556	0.063100	5.131E-19	18.185925

TABLE 3. Result comparisons of algorithms for low-dimensional functions.

Function	Index	SDO	PSO	GA	DE	CS	GSA	ABC
$f_{14}(x)$	Mean	0.998004	0.998004	3.854855	0.998004	0.998004	3.472815	0.998005
	Std	0	0.000000	2.766177	0	0	2.529209	6.916E-06
$f_{15}(x)$	Mean	3.075E-04	3.666E-04	0.001175	3.075E-04	3.075E-04	0.002358	5.097E-04
	Std	9.553E-20	1.908E-04	0.001501	1.420E-19	3.917E-08	0.001143	4.988E-05
$f_{16}(x)$	Mean	-1.031628	-1.031628	-1.031628	-1.031628	-1.031628	-1.031628	-1.031628
	Std	2.278E-16	6.775E-16	4.277E-10	6.775E-16	6.775E-16	5.532E-16	6.649E-16
$f_{17}(x)$	Mean	0.397887	0.397887	0.397887	0.397887	0.397887	0.397887	0.397887
	Std	0	0	1.225E-08	0	0	0	6.751E-10
$f_{18}(x)$	Mean	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000
	Std	6.027E-16	2.188E-15	1.837E-07	2.030E-15	1.882E-15	1.690E-15	2.861E-11
$f_{19}(x)$	Mean	-3.862782	-3.862782	-3.862782	-3.862782	-3.862782	-3.862782	-3.862782
	Std	2.278E-15	2.710E-15	1.454E-09	2.710E-15	2.710E-15	2.449E-15	1.668E-15
$f_{20}(x)$	Mean	-3.31085	-3.254072	-3.298217	-3.286327	-3.321995	-3.321995	-3.321995
	Std	4.886E-16	0.060415	0.048370	0.055415	1.917E-13	1.355E-15	1.525E-15
$f_{21}(x)$	Mean	-10.153200	-6.131172	-6.993384	-9.984785	-1.015E+01	-6.975879	-10.110816
	Std	3.645E-15	2.835887	3.699551	0.922443	3.803E-14	3.548979	0.168284
$f_{22}(x)$	Mean	-10.402941	-8.217934	-9.225009	-10.402941	-10.402941	-10.402941	-10.402941
	Std	2.967E-15	2.978914	2.686254	1.745E-15	2.795E-14	0.000000	3.529E-14
$f_{23}(x)$	Mean	-10.536410	-7.716923	-9.340499	-10.536410	-10.536410	-10.536410	-10.536410
	Std	1.954E-15	3.250563	2.731154	1.807E-15	4.200E-12	1.682E-15	8.095E-14

GA and DE on function f_{20}. The results reveal that SDO shows a distinct advantage in terms of exploration. This is owing to the fact that the exploration mechanism is integrated into SDO.

E. ANALYSIS OF AVOIDANCE OF LOCAL OPTIMA

It is very difficult for algorithms to find the optimal solutions of composite functions because they are required to balance between exploration and exploitation. Therefore, composite functions are very suitable for assessing local optima avoidance which is able to well balance exploration and exploitation. Table 4 shows the optimization results of above-mentioned algorithms on composite functions. These results prove that SDO algorithm is significantly effective in balancing exploration and exploitation. Such a merit originated from an adaptive mechanism is employed to the update of search agents in SDO algorithm: the early iterations tend to be dedicated to exploration ($|L| > 1$) while the later iterations to exploitation ($|L| < 1$).

F. ANALYSIS OF CONVERGENCE BEHAVIOR

Figs. 8-11 show the convergence curves of SDO and other algorithms on different types of functions. Obviously, SDO is very competitive with other metheuristic methods with

TABLE 4. Result comparisons of algorithms for composition functions.

Function	Index	SDO	PSO	GA	DE	CS	GSA	ABC
$f_{24}(x)$	Mean	8.699E-11	98.536796	80.000017	3.298E-04	0.322252	1.420E-17	16.358285
	Std	1.111E-10	101.571907	44.721368	0.0152988	0.577046	5.439E-18	35.404507
$f_{25}(x)$	Mean	6.362105	126.37533	199.33855	22.009693	10.208335	200	105.13300
	Std	4.069897	36.780146	90.539010	8.3230419	2.951655	47.140452	28.048742
$f_{26}(x)$	Mean	164.847875	327.81687	476.32103	184.88562	199.36633	172.36085	281.55826
	Std	57.3911974	55.592703	91.317600	34.802273	39.637477	82.861637	48.081802
$f_{27}(x)$	Mean	299.384752	438.28038	577.10983	349.45830	329.10352	333.620196	331.11361
	Std	34.203724	25.953010	43.893813	140.06337	13.329504	154.634938	20.376096
$f_{28}(x)$	Mean	2.204867	60.922190	211.919849	12.579056	10.458422	225.066682	21.134341
	Std	1.845317	37.765887	106.208138	3.4039261	1.953953	86.433716	12.409218
$f_{29}(x)$	Mean	500.150184	636.57241	822.50299	512.06662	505.75894	772.80849	527.71376
	Std	0.194042	184.41463	179.61784	44.758834	1.274708	65.389633	24.504563

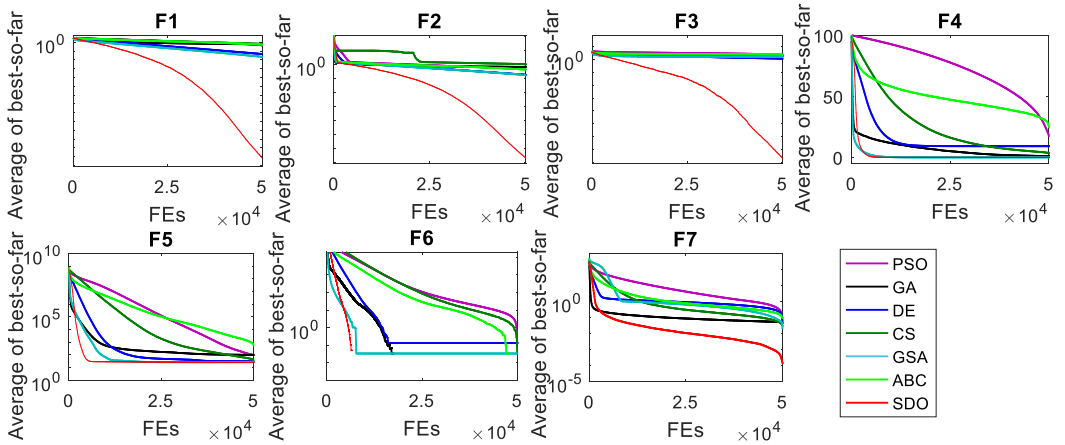

FIGURE 8. Convergence comparisons of algorithms for unimodal functions.

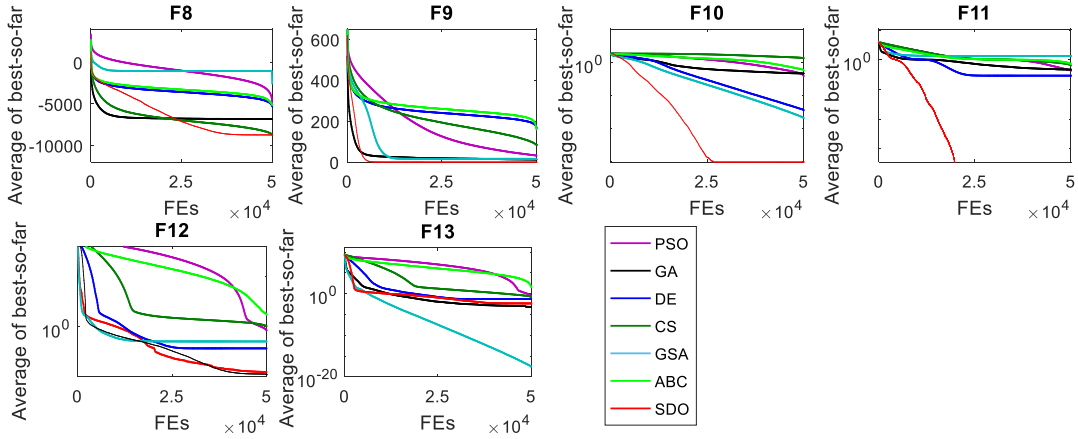

FIGURE 9. Convergence comparisons of algorithms for multimodal functions.

respect to convergence rate. SDO shows two different convergence behaviors [41]. First, SDO is subject to be accelerated quickly from the initial stage of iterations to the final stage of iterations and then converges to the global optimum (i.e., f_1, f_2, f_3). This is owing to that SDO can find the region with the optimal solution in the early stage of iterations and has an excellent local search ability. Second, SDO tends to converges towards the optimal solution only in the later

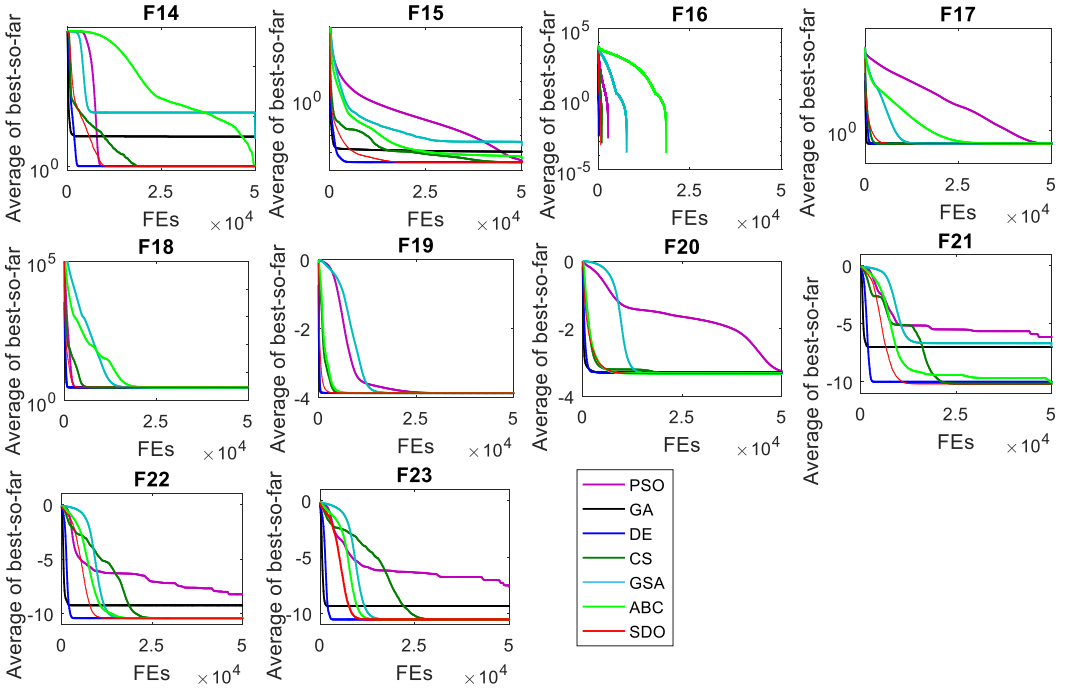

FIGURE 10. Convergence comparisons of algorithms for low-dimensional functions.

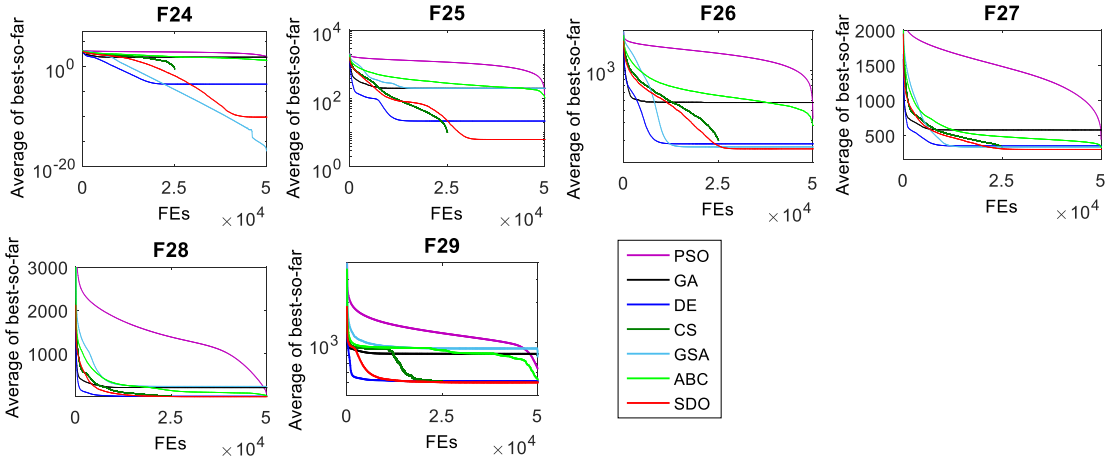

FIGURE 11. Convergence comparisons of algorithms for composite functions.

stage, this behavior probably results from that SDO fails to find the promising region in the early stage while after multiple attempts it finds the promising region with success in the second half of iterations (i.e., $f_8, f_{12}, f_{15}, f_{24}, f_{25}, f_{27}$). These results denote that SDO has a good ability to explore the global optimum and avoid the local optima. Evidently, these convergence curves reveal that SDO tends to adopt an adaptive search mechanism to different functions, demonstrating its better success in converging to the global optima for different mathematical problems.

G. ANALYSIS OF SCALABILITY

Since many optimization problems in the real world have a considerable number of variables, it is very important for us to analyze the scalability of the algorithm. Both unimodal and multimodal functions with a different number of dimensions are employed in this experiment. The number of dimensions is increased from 40 to 200 with the step of 10. Fig. 12 shows the scalability comparisons of SDO and other algorithms on these employed functions. It is obvious that SDO degrades much more slowly than the other algorithms on most of the scalable functions as the number of dimensions increases. For function f_8, although SDO degrades more quickly than GA and DE when the number of dimensions is less than 120, however after that, this degradation of SDO is improved. These figures discover that increasing the number of dimensions of variable space has the least influence on the performance of SDO.

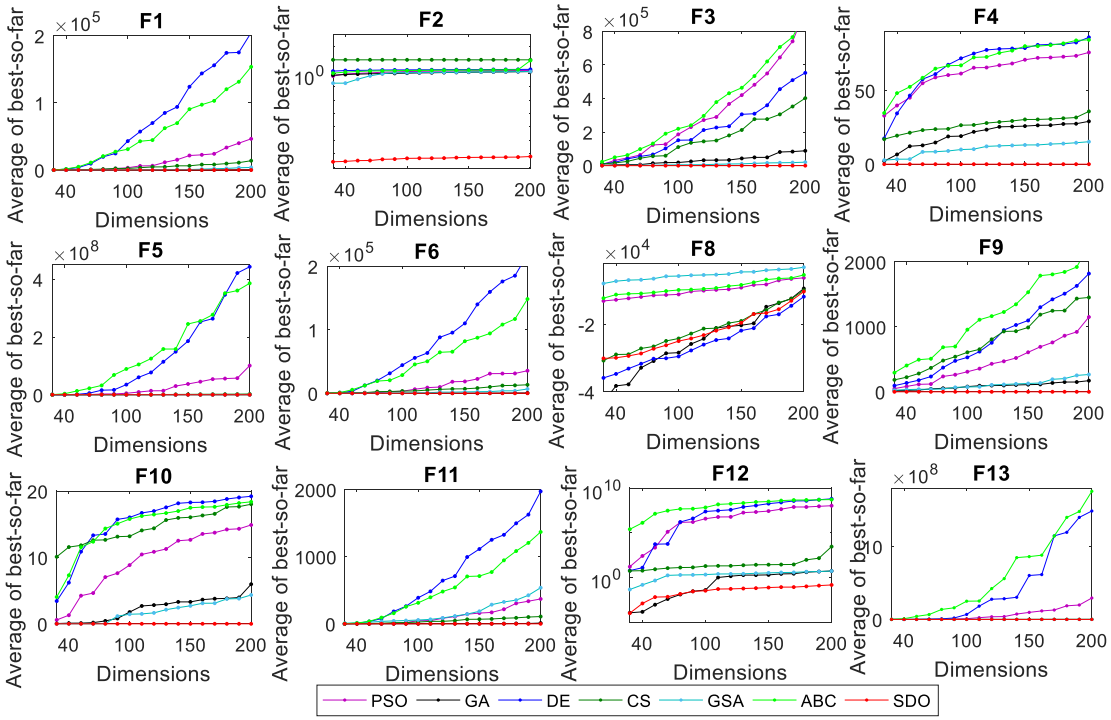

FIGURE 12. Scalability comparisons of algorithms for unimodal and multimodal functions.

H. ANALYSIS OF STATISITICAL SIGNIFICANCE

Wilcoxon Signed-Rank Test (WSRT) can effectively evaluate the overall performance of an algorithm. The test of WSRT with 95% significance level ($a = 0.05$) on 29 benchmark functions in 30 runs is implemented. The analysis for the test results is listed in Tables 5 and 6, respectively, where '=' indicates that there is no statistically significant difference between SDO and the comparative algorithm, '+' indicates that the null hypothesis will be rejected and SDO performs better than the comparative one and '-' vice versa. The sum of '+', '-' and '=' for each function in 30 runs is summarized in Table 7. As we observe in these tables, the results show that SDO has a much better performance at 95% significance level than its competitors on these four types of test functions.

IV. CONSTRAINT ENGINEERING PROBLEMS USING SDO

To further verify the performance of SDO algorithm, six constrained engineering problems are employed in this subsection. These constrained engineering cases are three-bar truss design, cantilever beam design, tension/compression spring design, rolling element bearing design, belleville spring design, and hydrostatic thrust bearing design. They have different constraints with different natures, thus a constraint handling method needs to be utilized. There are different constraint handling methods: static penalty, dynamic penalty, adaptive penalty, annealing penalty, co-evolutionary penalty, and death penalty. Of those, the most frequently used one is the penalty function because of its simplicity and less computational costs. Additionally, this method is not subject to making use of the information of infeasible solutions. Therefore, SDO algorithm is equipped with the death penalty function to handle constraints in these engineering problems. The optimization formulations of these 6 engineering problems are given in Appendix B.

A. THREE-BAR TRUSS DESIGN

This case, taken from [66] depicted in Fig. 13, needs to deal with a statically loaded three-bar truss with the minimum weight. It has three constraints on stress, deflection, and buckling and two variables (x_1 and x_2) to adjust the sectional areas. This case is very popular owing to its difficult constrained variable space.

This case was employed by many scholars with some metaheuristic methods in literature: SC [50], PSO-DE [67], DEDS [68], HEAA [69], and CS [70]. Our method is compared with these works and the comparisons of their results are provided in Table 8. The results of these algorithms report that SDO provides very competitive results in terms of different indices with the same or less computational costs compared to the other algorithms. The comparisons of the best solutions offered by considered methods are provided in Table 9. SDO is able to find the best optimal design of all the considered optimizers. Also, it can be seen that although the weights of three-bar truss offered by SDO, PSO-DE, DEDS, and HEAA are equal, the obtained design variables among them are different. Therefore, this suggests that our algorithm can provide a new best optimal design for this case. We give the function value and each constraint value

TABLE 5. Statistical comparisons of WSRT for SDO vs GA, PSO, and DE.

Function	SDO vs PSO				SDO vs GA				SDO vs DE			
	p-value	T-	T+	Winner	*p*-value	T-	T+	Winner	p-value	T-	T+	Winner
$f_1(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_2(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_3(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_4(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_5(x)$	4.73E-06	10	455	-	2.13E-06	2	463	-	1.73E-06	0	465	-
$f_6(x)$	0.2552	0	465	=	1	0	465	=	1.22E-04	0	465	-
$f_7(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_8(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	5.32E-03	368	97	+
$f_9(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_{10}(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_{11}(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_{12}(x)$	1.73E-06	0	465	-	1.64E-05	23	442	-	1.73E-06	0	465	-
$f_{13}(x)$	9.71E-05	43	422	-	6.27E-02	323	142	=	4.53E-04	62	403	-
$f_{14}(x)$	1	0	465	=	1.22E-05	0	465	-	1.56E-02	0	465	-
$f_{15}(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_{16}(x)$	1	0	465	=	1.25E-01	0	465	=	3.13E-02	0	465	-
$f_{17}(x)$	1	0	465	=	3.91E-03	0	465	-	1	0	465	=
$f_{18}(x)$	7.81E-03	62	403	-	1.99E-03	15	450	-	4.90E-02	84	381	-
$f_{19}(x)$	1	0	465	=	3.79E-06	0	465	-	1	0	465	=
$f_{20}(x)$	1.96E-04	0	465	-	1.73E-06	0	465	-	9.78E-05	0	465	-
$f_{21}(x)$	9.90E-06	0	465	-	3.71E-06	0	465	-	1.45E-07	0	465	-
$f_{22}(x)$	3.40E-04	4	461	-	1.73E-06	0	465	-	2.44E-04	0	465	-
$f_{23}(x)$	2.37E-04	1.5	463.5	-	1.71E-06	0	465	-	9.63E-07	0	465	-
$f_{24}(x)$	6.33E-04	0	465	-	7.52E-04	0	465	-	0.3125	0	465	-
$f_{25}(x)$	1.96E-03	0	465	-	5.90E-03	0	465	-	8.39E-3	136	329	-
$f_{26}(x)$	1.40E-03	4	461	-	2.83E-04	0	465	-	0.38526	260	205	=
$f_{27}(x)$	1.67E-02	28	437	-	6.28E-03	0	465	-	0.068231	145	320	=
$f_{28}(x)$	6.32E-04	15	450	-	2.91E-04	0	465	-	0.55217	210	255	-
$f_{29}(x)$	0.06822	206	259	-	3.02E-02	124	341	-	0.1832	206	259	=

with respect to FEs in Fig. 14, in which SDO offers a good convergence rate for this problem.

B. CANTILEVER BEAM DESIGN

As illustrated in Fig. 15, the cantilever beam is composed of 5 hollow square blocks with constant thickness [71]. The beam is rigidly supported at the first block and there is vertical force acting at the free end of the fifth block. So the objective of this case needs to minimize the weight of the beam and meanwhile only meet the constraint requirement on an upper limit on the vertical displacement of the free end. There are five decision variables that are respectively lengths of the different blocks.

This problem is attempted using our method and many reported metaheuristics, including SOS [72], CS [70], MMA [71], GCA-I [71], GCA-II [71] and MFO [73], and the comparisons of their results are provided in Table 10. Overall, the results of SDO have an obvious advantage.

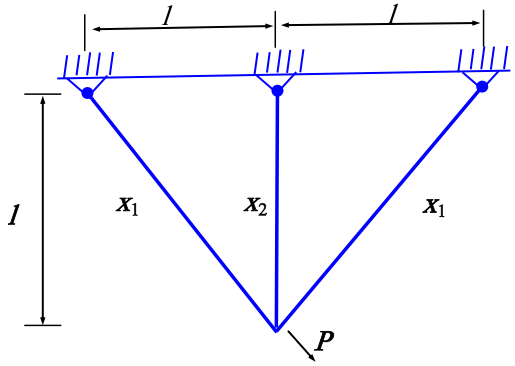

FIGURE 13. Three-bar truss design problem.

TABLE 6. Statistical comparisons of WSRT for SDO vs CS, GSA, and ABC.

Function	SDO vs CS				SDO vs GSA				SDO vs ABC			
	p-value	T-	T+	Winner	*p*-value	T-	T+	Winner	p-value	T-	T+	Winner
$f_1(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_2(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_3(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_4(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_5(x)$	1.73E-06	0	465	-	2.88E-06	5	460	-	1.73E-06	0	465	-
$f_6(x)$	1	0	465	=	1	0	465	=	1	0	465	=
$f_7(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$F_8(x)$	3.71E-01	189	276	=	1.73E-06	0	465	-	1.73E-06	0	465	-
$F_9(x)$	1.73E-06	0	465	-	1.72E-06	0	465	-	1.73E-06	0	465	-
$f_{10}(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_{11}(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_{12}(x)$	1.73E-06	0	465	-	0.64416	210	255	=	1.73E-06	0	465	-
$f_{13}(x)$	1.64E-05	23	442	-	1.73E-06	465	0	+	1.73E-06	0	465	-
$f_{14}(x)$	1	0	465	=	2.56E-06	0	465	-	1.73E-06	0	465	-
$f_{15}(x)$	1.73E-06	0	465	-	1.73E-06	0	465	-	1.73E-06	0	465	-
$f_{16}(x)$	1	0	465	=	2.21E-05	0	465	-	5.00E-01	0	465	=
$f_{17}(x)$	1	0	465	=	1	0	465	=	1.73E-06	0	465	-
$f_{18}(x)$	4.45E-04	134.5	330.5	-	5.80E-06	1.5	463.5	-	1.73E-06	0	465	-
$f_{19}(x)$	1	0	465	=	2.21E-05	0	465	-	5.99E-07	0	465	-
$f_{20}(x)$	1.73E-06	0	465	-	5.00E-01	0	465	=	1.95E-03	0	465	-
$f_{21}(x)$	9.57E-07	0	465	-	1.71E-06	0	465	-	1.71E-06	0	465	-
$f_{22}(x)$	4.54E-05	5.5	459.5	-	2.44E-04	0	465	-	1.73E-06	0	465	-
$f_{23}(x)$	1.68E-06	0	465	-	5.73E-07	0	465	-	1.73E-06	0	465	-
$f_{24}(x)$	3.26E-03	0	465	-	8.27E-04	465	0	+	8.15E-05	0	465	-
$f_{25}(x)$	5.82E-04	60	405	-	1.97E-03	0	465	-	3.17E-03	0	465	-
$f_{26}(x)$	0.41853	244	221	=	5.11E-03	91	374	-	3.79E-05	6	459	-
$f_{27}(x)$	0.41091	275	190	=	0.32628	221	244	=	0.12526	275	190	=
$f_{28}(x)$	0.03271	156	309	-	2.65E-04	51	414	-	2.88E-06	5	460	--
$f_{29}(x)$	9.16E-02	146	319	=	3.51E-04	91	374	-	0.28162	190	275	=

TABLE 7. Statistical results of WSRT for SDO.

Function types	SDO vs PSO (+/=/-)	SDO vs GA (+/=/-)	SDO vs DE (+/=/-)	SDO vs CS (+/=/-)	SDO vs GSA (+/=/-)	SDO vs ABC (+/=/-)
Unimodal	6/1/0	6/1/0	7/0/0	6/1/0	6/1/0	6/1/0
Multimodal	6/0/0	5/1/0	5/0/1	5/1/0	4/1/1	6/0/0
Fixed-dimensional	6/4/0	9/1/0	8/2/0	6/4/0	8/2/0	9/1/0
Composite	6/0/0	6/0/0	3/3/0	3/3/0	4/1/1	4/2/0
Total	24/5/0	26/3/0	23/5/1	20/9/0	22/5/2	25/4/0

In terms of the 'Best' index, SDO provides the best results compared to the others. Meanwhile, the results provided by SDO are very close to those provided by SOS in terms of the 'Mean' index. The comparisons of the best solutions offered by reported algorithms are provided in Table 11. From this table, the results reveal that SDO is able to obtain the best optimal design for this problem. Fig. 16 shows the function value and constraint value with respect to FEs.

C. TENSION/COMPRESSION SPRING DESIGN

This problem, originated from [74], needs to minimize the weight of a tension/compression spring with respect to the

TABLE 8. Result comparisons in literature for three-bar truss design. "NA" stands for not available.

Methods	Worst	Mean	Best	Std	FEs
SC	263.969756	263.903356	263.895846	1.3E-02	17,610
PSO-DE	263.895843	263.895843	263.895843	4.5E-10	17,600
DEDS	263.895849	263.895843	263.895843	9.7E-07	15,000
HEAA	263.896099	263.895865	263.895843	4.9E-05	15,000
CS	NA	264.0669	263.97156	9.0E-05	15,000
SDO	2.63895847	2.63895845	263.895843	1.12E-06	15,000

TABLE 9. Comparisons of best solutions offered by reported optimizers for three-bar truss design.

	SC	PSO-DE	DEDS	HEAA	CS	SDO
x_1	0.788621	0.7886751	0.788675	0.788680	0.788670	0.788698
x_2	0.408401	0.4082482	0.408248	0.408234	0.409020	0.408184
g_1	NA	-5.29E-11	-1.77E-08	NA	-0.00029	-8.052E-10
g_2	NA	-1.4637475	-1.464101	NA	-0.26853	-1.464174
g_3	NA	-0.5362524	-0.535898	NA	-0.73176	-0.535825
f_1	263.895847	263.895843	263.895843	263.895843	263.971623	263.895843

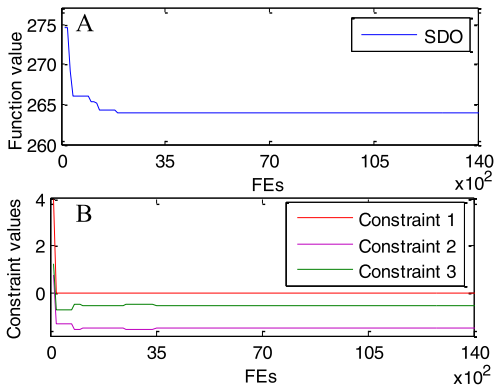

FIGURE 14. Function value and constraint values versus FEs for three-bar truss design.

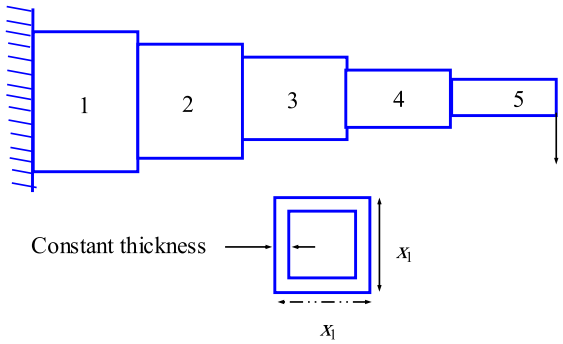

FIGURE 15. Cantilever beam design problem.

constraints on shear stress, deflection, and surge frequency shown in Fig. 17. The decision variables of this problem are wire diameter (d), mean coil diameter (D), and number of active coils (N).

TABLE 10. Result comparisons in literature for cantilever beam design.

Methods	Worst	Mean	Best	Std	FEs
SOS	NA	1.33997	1.33996	1.1E-05	15,000
CS	NA	NA	1.33999	NA	125,000
MMA	NA	NA	1.3400	NA	NA
GCA-I	NA	NA	1.3400	NA	NA
GCA-II	NA	NA	1.3400	NA	NA
MFO	NA	NA	3.399880	NA	NA
SDO	1.340513	1.34009	1.339963	2.626E-05	15,000

TABLE 11. Comparisons of best solutions offered by reported optimizers for cantilever beam design.

	SOS	CS	MMA	GCA-I	GCA-II	MFO	SDO
x_1	6.01878	6.0089	6.0100	6.0100	6.0100	5.984871	6.021609
x_2	5.30344	5.3049	5.3000	5.3000	5.3000	5.316726	5.294696
x_3	4.49587	4.5023	4.4900	4.4900	4.4900	4.497332	4.500152
x_4	3.49896	3.5077	3.4900	3.4900	3.4900	3.513616	3.505450
x_5	2.15564	2.1504	2.1500	2.1500	2.1500	2.1616200	2.151867
g_1	NA	NA	NA	NA	NA	NA	-1.547E-9
f_2	1.33996	1.33999	1.3400	1.3400	1.3400	1.339988	1.339963

This problem is tackled by our algorithm and some other metaheuristic methods, such as GA2 [75], GA3 [76], CA [77], CPSO [78], HPSO [79], PSO2 [80], QPSO [80], UPSO [81], CDE [82], SSB [50], and $(\mu + \lambda)$ES [83]. The comparisons of their best results are shown in Table 12.

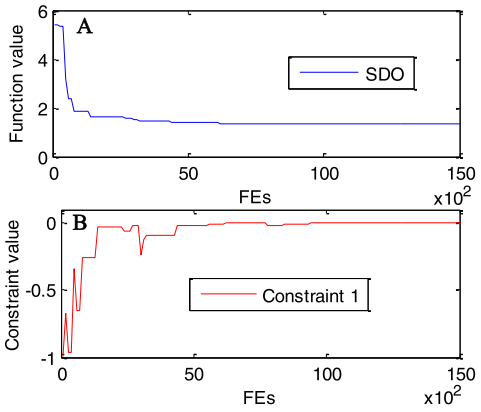

FIGURE 16. Function value and constraint value versus FEs for cantilever beam design.

TABLE 12. Result comparisons in literature for tension/compression spring design.

Methods	Worst	Mean	Best	Std	FEs
GA2	0.0128221	0.0127690	0.0127047	3.9390 E-05	900,000
GA3	0.0129730	0.0127420	0.0126810	5.9000 E-05	80,000
CA	0.0151156	0.0135681	0.0127210	8.4215 E-04	50,000
CPSO	0.0129240	0.0127300	0.0126747	5.1985 E-04	200,000
HPSO	0.0127191	0.0127072	0.0126652	1.5824 E-05	75,000
PSO2	0.0718020	0.0195550	0.0128570	0.0116620	2000
QPSO	0.0181270	0.0138540	0.0126690	1.3410E-03	2000
UPSO	0.0503651	0.0229478	0.0131200	7.2057E-03	10,000
CDE	0.0127900	0.0127030	0.0126702	2.7000E-05	240,000
SSB	0.0167173	0.0129227	0.0126692	5.9000E-04	25,167
(μ+λ)ES	NA	0.0131650	0.0126890	3.9000E-04	30,000
SDO	0.0126828	0.0126724	0.0126663	6.1899E-06	20,000

As shown in this table, SDO outperforms the others for finding the best optimal solutions in terms of the 'Worst', 'Mean' and 'Std' indexes. The comparisons of the best solutions offered by reported algorithms are provided in Table 13. It has been noticed that SDO is able to find another promising design for this problem. The function value and each constraint value with respect to FEs are depicted in Fig. 18.

D. ROLLING ELEMENT BEARING DESIGN

This problem [85], [86] is to maximize the dynamic load carrying capacity of rolling element bearing as illustrated in Fig. 19. The maximization process is subject to constraints on the geometry and kinematics. This problem has ten decision variables: pitch diameter (D_m), ball diameter (D_b), number of balls (Z), inner and outer raceway curvature coefficients (f_i and f_o), and other internal geometry parameters (K_{Dmin}, K_{Dmax},ε, e, and ζ) that only appear in constraints.

The problem was attempted by GA4 [84], TLBO [86], ABC [86], and MBA [87], the best results of these approaches

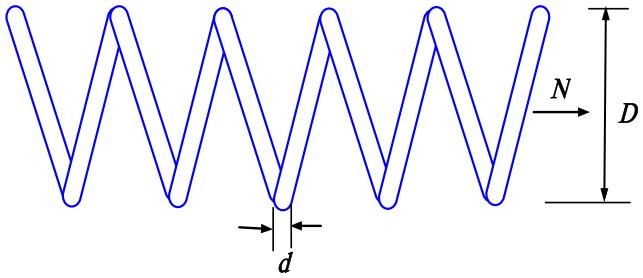

FIGURE 17. Tension/compression spring design problem.

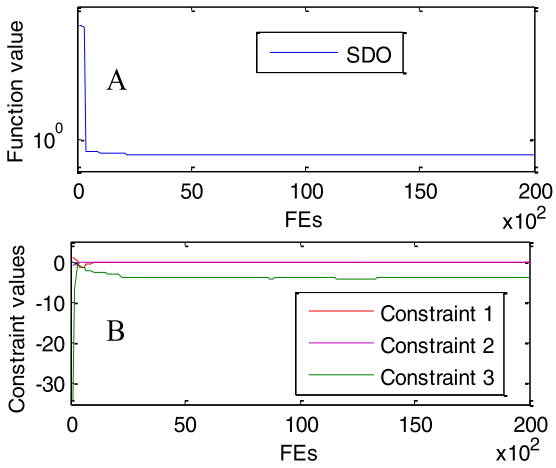

FIGURE 18. Function value and constraint values versus FEs for tension/compression spring.

and SDO are compared in Table 14. It can be seen that the proposed method is able to find the best optimal solutions in terms of the 'Worst', 'Mean', 'Best', 'Std' indexes over the other algorithms. Although the number of function evaluations of SDO is more than those of both ABC and TLBO, the overall results show that the performance of SDO is considerably better that of both algorithms for this problem. Table 15 represents the comparisons of best optimal solutions for the considered algorithms in terms of design decision variables and function values. This table reveals that SDO obtains a design with the maximum dynamic load carrying capacity. Fig. 20 shows the function value and each constraint value with respect to FEs.

E. BELLEVILLE SPRING DESIGN

The objective of this design [88], as shown in Fig. 21, is to minimize the weight of belleville spring. These constraints consist of compressive stress, deflection, height to deflection, height to maximum height, outer diameter, inner diameter, and slope. There are four decision variables to be optimized such as thickness of the spring (t), height (h) of the spring, internal diameter of the spring (D_i), and external diameter of the spring (D_e).

The problem was solved by GA5 [88], GeneAS [89], Siddal [90], ABC [86] and TLBO [86], and the best solutions of various methods for this case are provided in Table 16.

TABLE 13. Comparisons of best solutions offered by reported optimizers for tension/compression spring design.

	GA3	CA	CPSO	HPSO	CDE	(μ+λ)ES	SDO
$x_1(d)$	0.051989	0.050000	0.051728	0.051706	0.051609	0.052836	0.0518977
$x_2(D)$	0.363965	0.317395	0.357644	0.357126	0.354714	0.384942	0.3617523
$x_3(N)$	10.890522	14.031795	11.244543	11.265083	11.410831	9.807729	10.7479462
g_1	−0.000013	0	-8.25E-04	-3.06E-06	−0.000039	-0.000001	-2.435E-06
g_2	−0.000021	-0.000075	-2.52E-05	- 1.39E-06	- −0.000183	0	-1.137E-05
g_3	−4.061338	-3.967960	-4.051306	-4.054583	−4.048627	-4.106146	-4.0635417
g_4	−0.722698	-0.755070	-0.727085	-0.727445	−0.729118	-0.708148	-0.7242333
f_3	0.0126810	0.0127210	0.0126747	0.0126652	0.0126702	0.012689	0.0126663

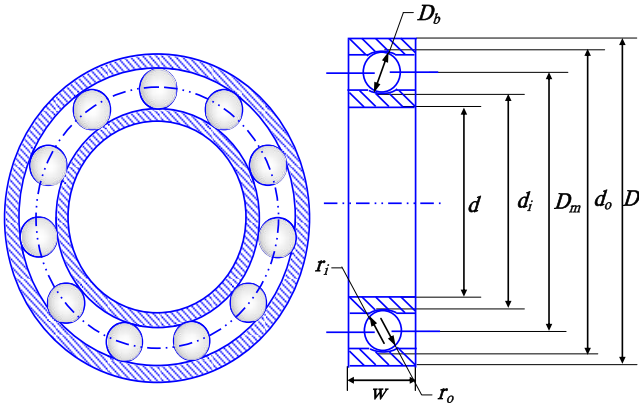

FIGURE 19. Rolling element bearing design.

TABLE 14. Result comparisons in literature for rolling element bearing design.

Methods	Worst	Mean	Best	Std	FEs
GA4	NA	NA	81843.3000	NA	225,000
ABC	78897.8100	81496.0000	81859.7416	NA	10,000
TLBO	80807.8551	81438.9870	81859.7400	NA	10,000
MBA	84440.1948	85321.4030	85535.9611	211.5200	15,100
SDO	85144.1639	85443.6665	8.5538.85468	112.2171	15,000

As shown in this table, SDO obtains the same results as both methods in terms of the 'Best' index with less computational efforts. However, the results provided by SDO are better than those provided by ABC in terms of the 'Worst' and 'Mean' indexes. The comparisons of the best solutions offered by reported algorithms are provided in Table 17. From this table, SDO is able to offer a similar design compared to TLBO. This is the best optimal solution obtained so far for this problem. Fig. 22 shows the convergence rate of SDO for finding the best optimal solution.

F. HYDROSTATIC THRUST BEARING DESIGN

The last utilized engineering problem [90], as shown in Fig. 23, is the hydrostatic thrust bearing design problem.

TABLE 15. Comparisons of best solutions offered by reported optimizers for rolling element bearing design.

	GA4	TLBO	MBA	SDO
$x_1(D_m)$	125.7171	125.7191	125.7153	1.257E+02
$x_2(\mathrm{D_b})$	21.423	21.42559	21.423300	21.424905
$x_3(Z)$	11	11	11.000	11.430211
$x_4(f_i)$	0.515	0.515	0.515000	0.515002
$x_5(f_o)$	0.515	0.515	0.515000	0.519304
$x_6(K_{\mathrm{Dmin}})$	0.4159	0.424266	0.488805	0.487755
$x_7(K_{\mathrm{Dmax}})$	0.651	0.633948	0.627829	0.629992
$x_8\ (\varepsilon)$	0.300043	0.3	0.300149	0.300039
$x_9(e)$	0.0223	0.068858	0.097305	5.351E-02
$x_{10}(\zeta)$	0.751	0.799498	0.646095	0.665982
g_1	0.000821	0	0	1.107E-04
g_2	13.732999	13.15257	8.630183	8.706977
g_3	2.724000	1.5252	1.101429	1.249638
g_4	3.606000	0.719056	2.040448	14.094788
g_5	0.717000	16.49544	0.715366	12.660371
g_6	4.857899	0	23.611002	0.717208
g_7	0.003050	0	0.000480	6.393E-04
g_8	0.000007	2.559363	0	1.445455
g_9	0.000007	0	0	2.076E-06
g_{10}	0.000005	0	0	4.304E-03
f_4	81843.3	81859.74	85535.9611	85538.85468

TABLE 16. Result comparisons in literature for belleville spring design.

Mean	Best	Std	FEs
1.995475	1.979675	0.07	150,000
1.9796875	1.979675	NA	150,000
1.988263	1.979675	0.0032	50,000

The objective of this problem is to minimize the power loss subject to seven constraints on load-carrying capacity, inlet oil pressure, oil temperature rise, oil film thickness and some

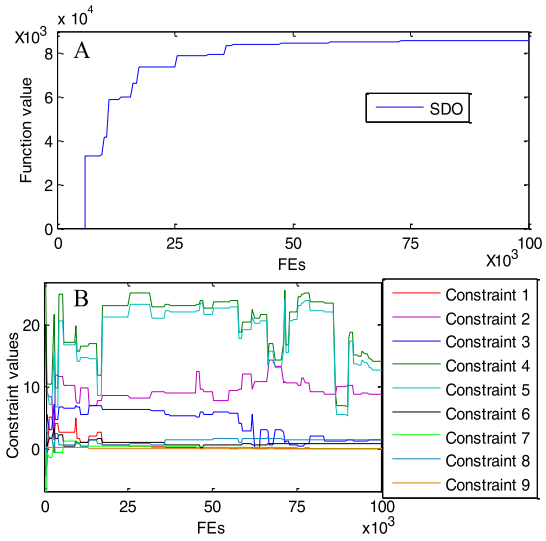

FIGURE 20. Function value and constraint values versus FEs for rolling element bearing design.

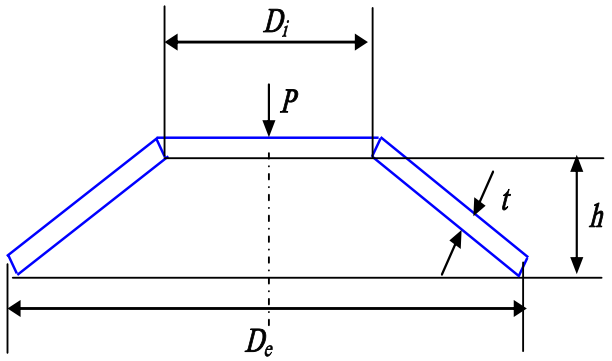

FIGURE 21. Belleville spring design.

TABLE 17. Comparisons of best solutions offered by reported optimizers for belleville spring design.

	GA5	GeneAS	Siddal	TLBO	SDO
t	0.208	0.205	0.204	0.204143	0.204143
h	0.2	0.201	0.2	0.2	0.200000
D_i	8.751	9.534	10.030	10.03047	10.030473
D_e	11.067	11.627	12.010	12.01	12.009999
g_1	2145.4109	-10.3396	134.0816	1.77E-06	-4.075E-10
g_2	39.75018	2.8062	-12.5328	7.46E-08	-7.832E-04
g_3	0	0.0010	0	5.8E-11	-1.399E-09
g_4	1.592	1.5940	1.5960	1.595857	-1.595857
g_5	0.943	0.3830	0	2.35E-09	-7.621E-07
g_6	2.316	2.0930	1.9800	1.979527	-1.979528
g_7	0.21364	0.20397	0.19899	0.198966	-0.198966
f_5	2.121964	2.01807	1.978715	1.979675	1.979675

physical requirements. The decision variables are bearing step radius (R), recess radius (R_o), oil viscosity (μ), and flow rate (Q).

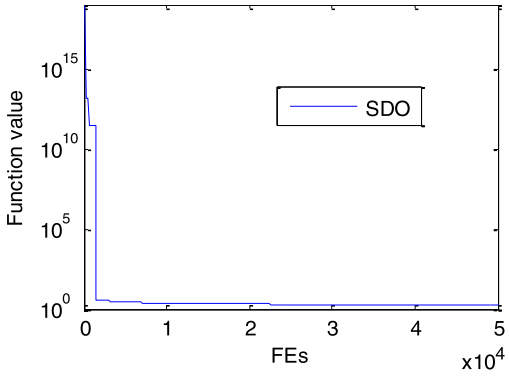

FIGURE 22. Convergence curve of SDO for cantilever beam design.

TABLE 18. Result comparisons in literature for hydrostatic thrust bearing design.

Methods	Worst	Mean	Best	Std	FEs
IPSO	NA	1757.3768400	1632.2149	16.851024	90,000
GASO	NA	NA	1950.2860	NA	16,000
TLBO	2096.80127	1797.70798	1625.443	NA	50,000
ABC	2144.836	1861.554	1625.44276	NA	50,000
Gene AS	NA	NA	2161.6	NA	NA
BGA	NA	NA	2295.1	NA	NA
SDO	2012.862318	1753.643271	1626.62227	116.3291	50,000

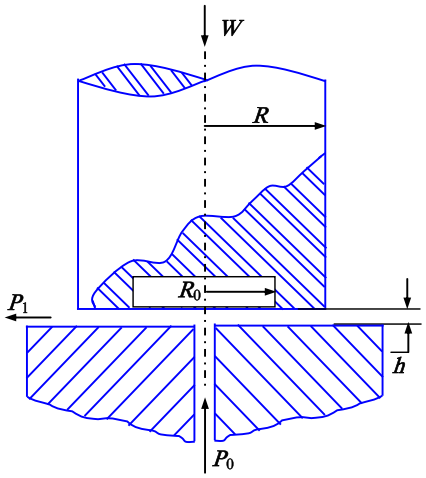

FIGURE 23. Hydrostatic thrust bearing design.

Table 13 provides the comparisons of the best results obtained from SDO and other optimizers such as IPSO [91], GASO [66], TLBO [86], ABC [86], GeneAS [89], and BGA [89]. As we see in this table, SDO surpasses the other optimizers for offering the best solutions in terms of the 'Worst' and 'Mean' indexes with less or equal computational costs. Besides, SDO provides a competitive result in terms of the 'Best' index compared to both TLBO and ABC. The comparisons of the best solutions offered by reported algorithms are provided in Table 19. This table detailedly lists the best optimal variables and the best optimal

TABLE 19. Comparisons of best solutions offered by reported optimizers for hydrostatic thrust bearing design.

	IPSO	GASO	GeneAS	BGA	TLBO	SDO
R	5.956868685	6.271	6.778	7.7077	5.95578050261541	5.957853282295
R_o	5.389175395	12.901	6.234	6.549	5.38901305194167	5.391201948055
μ	5.40213310	5.605	6.096	6.619	5.35869726E-06	5.361337511E-06
Q	2.30154678	2.938	3.809	4.849	2.26965597280973	2.273155235181
g_1	22.01094912	2126.86734	8329.7681	1440.6013	1.375E-04	6.80638998
g_2	0.00000000	68.0396	177.3527	297.1495	2.100E-08	0.684507284
g_3	0.58406092	3.705191	10.684543	17.353800	3.244E-04	0.0420416175
g_4	0.00033480	0.000559	0.000652	0.000891	0.566767	3.2530 E-04
g_5	0.56769329	0.666000	0.544000	0.528000	9.964E-04	0.5666513
g_6	0.00083138	0.000805	0.000717	0.000624	9.076E-06	-9.9635 E-04
g_7	7.61684431	849.718683	83.618221	467.686527	1.375E-04	0.516470143
f_6^*	1632.2149	1950.2860	2161.4215	2296.2119	1625.442764	1626.62227

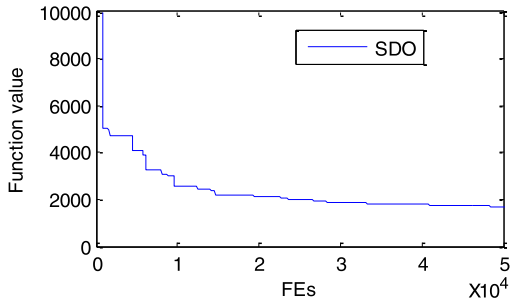

FIGURE 24. Convergence curve of SDO for hydrostatic thrust bearing design.

power loss offered by the above methods. The results indicate that SDO is superior to four of the optimizers and performs almost as well as TLBO. Fig. 24 shows that SDO is able to effectively converge to the global optimum for this problem.

V. CONCLUSIONS

The above study discover that SDO has an advantage over other metaheuristic methods. First, the results of unconstrained test functions show the effectiveness of SDO in terms of exploration, exploitation, local optima avoidance, and convergence rate. Second, the results of constrained engineering problems indicate the competiveness of our algorthm in terms of computational expense and covergence rate. Eventually, the study on a set of engineering problems suggests the application of SDO in challenging real-world problems.

The results on unconstrained functions show SDO is highly competitive with PSO, ABC, DE, GA, GSA, and CS from different aspects. The results on the unimodal functions reveal the high exploitation ability of SDO. The results on multimodal and fixed-dimensional functions prove the superior exploration ability of SDO. The results on composite functions suggest the excellent local optima avoidance of SDO. Despite, the good convergence rate of SDO is also confirmed on all the unconstrained functions. Moreover, the comprehensive results on constrained engineering problems evidence the success and effectiveness of SDO in solving real constrained problems.

There are several research directions for future work. First, the binary and multi-objective versions of SDO may be developed to solve complex discrete and multi-objective problems, respectively. Second, SDO may be equipped with some stochastic or evolutionary operators to enhance its optimization capability. Additionally, the SDO may be hybridized with other stochastic algorithms to improve its optimization capability. Eventually, SDO, its variants, and its hybridized versions may all be applied to optimization problems in different fields.

APPENDIX A

See Tables 20–23.

APPENDIX B

A. THREE-BAR TRUSS DESIGN

Consider variable $\vec{x} = [x_1, x_2]$.

$$\begin{aligned}
&\text{Minimize } f_1(\vec{x}) = (2\sqrt{2}\,x_2 + x_2) \times l.\\
&\text{Subject to } \mathrm{g}_1(\vec{x}) = \frac{\sqrt{2}\,x_1 + x_2}{\sqrt{2}\,x_1^2 + 2x_2x_1}P - \sigma \le 0,\\
&\qquad \mathrm{g}_2(\vec{x}) = \frac{x_2}{\sqrt{2}\,x_1^2 + 2x_2x_1}P - \sigma \le 0 \le 0,\\
&\qquad \mathrm{g}_3(\vec{x}) = \frac{x_2}{x_1 + \sqrt{2}\,x_2}P - \sigma \le 0.
\end{aligned}$$

where

$$l = 10\text{cm}, \quad P = 2KN/\text{cm}^2, \ \sigma = 2KN/\text{cm}^2.$$

Variable range $0 \le x_1 \le 1, 0 \le x_2 \le 1$.

TABLE 20. Unimodal test functions.

Name	Function	D	Range	f_{opt}
Sphere	$f_1(x)=\sum_{i=1}^{n}x_i^2$	30	$[-100,100]^D$	0
Schwefel 2.22	$f_2(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	$[-10,10]^D$	0
Schwefel 1.2	$f_3(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j)^2$	30	$[-100,100]^D$	0
Schwefel 2.21	$f_4(x)=\max_i\{\lvert x_i\rvert,1\le i\le n\}$	30	$[-100,100]^D$	0
Rosenbrock	$f_5(x)=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i)^2)+(x_i-1)^2)$	30	$[-30,30]^D$	0
Step	$f_6(x)=\sum_{i=1}^{n}(x_i+0.5)^2$	30	$[-100,100]^D$	0
Quartic	$f_7(x)=\sum_{i=1}^{n}ix_i^4+random[0,1)$	30	$[-1.28,1.28]^D$	0

TABLE 21. Multimodal test functions.

Name	Function	D	Range	f_{opt}
Schwefel	$f_8(x)=-\sum_{i=1}^{n}(x_i\sin(\sqrt{\lvert x_i\rvert}))$	30	$[-500,500]^D$	-12569.5
Rastrigin	$f_9(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)^2$	30	$[-5.12,5.12]^D$	0
Ackley	$f_{10}(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i)+20+e$	30	$[-32,32]^D$	0
Griewank	$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}(x_i-100)^2-\prod_{i=1}^{n}\cos(\frac{x_i-100}{\sqrt{i}})+1$	30	$[-600,600]^D$	0
Penalized	$f_{12}(x)=\frac{\pi}{n}\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_i+1)]$ $+(y_n-1)^2\}+\sum_{i=1}^{30}u(x_i,10,100,4)$	30	$[-50,50]^D$	0
Penalized2	$f_{13}(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{29}(x_i-1)^2\,p[1+\sin^2(3\pi x_{i+1})]$ $+(x_n-1)^2[1+\sin^2(2\pi x_{30})]\}+\sum_{i=1}^{30}u(x_i,5,10,4)$	30	$[-50,50]^D$	0

B. CANTILEVER BEAM DESIGN

Consider variable $\vec{x}=[x_1,x_2]$.

$$\text{Minimize } f_2(\vec{x})=0.0624(x_1+x_2+x_3+x_4+x_5).$$

$$\text{Subject to } g_1(\vec{x})=\frac{61}{x_1^3}+\frac{37}{x_2^3}+\frac{19}{x_3^3}+\frac{7}{x_4^3}+\frac{1}{x_5^3}-1\le 0,$$

Variable range $0.01\le x_i\le 100$, $i=1,\cdots,5$.

C. TENSION/COMPRESSION SPRING DESIGN

Consider variable $\vec{x}=[x_1,x_2,x_3]=[d,D,N]$.

$$\text{Minimize } f_3(\vec{x})=(x_3+2)x_2x_1^2.$$

$$\text{Subject to } g_1(\vec{x})=1-\frac{x_3x_2^3}{71785x_1^4}\le 0,$$

$$g_2(\vec{x})=\frac{4x_2^2-x_1x_2}{12566(x_2x_1^3-x_1^4)}+\frac{1}{5108x_1^2}-1\le 0,$$

$$g_3(\vec{x})=1-\frac{140.45x_1}{x_2^2x_3}\le 0,$$

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

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

D. ROLLING ELEMENT BEARING DESIGN

Consider variable $\vec{x}=[D_m,D_b,Z,f_i,f_o,K_{D\min},K_{D\max},\varepsilon,e,\zeta]$.

$$\text{Maximize } \begin{cases} f_4(\vec{x})=f_cZ^{2/3}D_b^{1.8} & \text{if } D_b\le 25.4mm \\ f_4(\vec{x})=3.647f_cZ^{2/3}D_b^{1.4} & \text{if } D_b>25.4mm \end{cases}.$$

$$\text{Subject to } g_1(\vec{x})=\frac{\phi_o}{2\sin^{-1}(D_b/D_m)}-Z+1\ge 0,$$

$$g_2(\vec{x})=2D_b-K_{D\min}(D-d)\ge 0,$$

$$g_3(\vec{x})=K_{D\max}(D-d)-2D_b\ge 0,$$

$$g_4(\vec{x})=D_m-(0.5-e)(D+d)\ge 0,$$

$$g_5(\vec{x})=(0.5+e)(D+d)-D_m\ge 0,$$

$$g_6(\vec{x})=D_m-0.5(D+d)\ge 0,$$

$$g_7(\vec{x})=0.5(D-D_m-D_b)-\varepsilon D_b\ge 0,$$

$$g_8(\vec{x})=\zeta B_w-D_b\le 0,$$

$$g_9(\vec{x})=f_i\ge 0.515,$$

$$g_{10}(\vec{x})=f_o\ge 0.515.$$

TABLE 22. Low-dimensional multimodal test functions.

Name	Function	D	Range	f_{opt}
Foxholes	$f_{14}(x)=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{j=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$	2	$[-65.536, 65.536]^D$	0.998
Kowalik	$f_{15}(x)=\sum_{i=1}^{11}\left\lvert a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right\rvert^2$	4	$[-5, 5]^D$	3.075×10^{-4}
Six Hump Camel	$f_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	$[-5, 5]^D$	-1.0316
Branin	$f_{17}(x)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10$	2	$[-5, 10]\times[0, 15]$	0.398
GoldStein-Price	$f_{18}(x)=\left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right]$ $\times\left[30+(2x_1+1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]$	2	$[-2, 2]^D$	3
Hartman 3	$f_{19}(x)=-\sum_{i=1}^{4}\exp\left[-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2\right]$	3	$[0, 1]^D$	-3.86
Hartman 6	$f_{20}(x)=-\sum_{i=1}^{4}\exp\left[-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2\right]$	6	$[0, 1]^D$	-3.322
Shekel 5	$f_{21}(x)=-\sum_{i=1}^{5}\left\lvert(x_i-a_i)(x_i-a_i)^T+c_i\right\rvert^{-1}$	4	$[0, 10]^D$	-10.1532
Shekel 7	$f_{22}(x)=-\sum_{i=1}^{7}\left\lvert(x_i-a_i)(x_i-a_i)^T+c_i\right\rvert^{-1}$	4	$[0, 10]^D$	-10.4028
Shekel 10	$f_{23}(x)=-\sum_{i=1}^{10}\left\lvert(x_i-a_i)(x_i-a_i)^T+c_i\right\rvert^{-1}$	4	$[0, 10]^D$	-10.5363

where

$$f_c = 37.91\left[1+\left\{1.04\left(\frac{1-\gamma}{1+\gamma}\right)^{1.72}\times\left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0.4}\right\}^{10/3}\right]^{-0.3}\times\left(\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{f_o(1+\gamma)^{\frac{1}{3}}}\right)\left(\frac{2f_i}{2f_i-1}\right)^{0.41},$$

$$\gamma=\frac{D_b\cos\alpha}{D_m},\quad f_i=\frac{r_i}{D_b},\ f_o=\frac{r_o}{D_b},$$

$$\phi_o = 2\pi-2\cos^{-1}\times\left\{\frac{\{(D-d)/2-3(T/4)\}^2}{2\{(D-d)/2-3(T/4)\}\{D/2-(T/4)-D_b\}},\times\frac{\{D/2-(T/4)-D_b\}^2-\{d/2+(T/4)\}^2}{2\{(D-d)/2-3(T/4)\}\{D/2-(T/4)-D_b\}}\right\}$$

$$T=D-d-2D_b,$$

$$D=160,\ d=90,\ B_w=30,\ r_i=r_o=11.033.$$

Variable range

$$0.5(D+d)\le D_m\le 0.6(D+d),$$
$$0.15(D-d)\le D_b\le 0.45(D-d),\ 4\le Z\le 50,$$
$$0.515\le f_i\le 0.6,\ 0.515\le f_o\le 0.6,$$
$$0.4\le K_{D\min}\le 0.5,\ 0.6\le K_{D\max}\le 0.7,$$
$$0.3\le\varepsilon\le 0.4,\ 0.02\le e\le 0.1,\ 0.6\le\zeta\le 0.85.$$

E. BELLEVILLE SPRING DESIGN

Consider variable $\vec{x}=[t, h, D_i, D_e]$.

Minimize: $f_5(\vec{x})=0.07075\pi\left(D_e^2-D_i^2\right)t.$

Subject to $g_1(\vec{x})=S-\frac{4E\delta_{\max}}{(1-\mu^2)\alpha D_e^2}\left[\beta(h-\frac{\delta_{\max}}{2})+\gamma t\right]\ge 0,$

$$g_2(\vec{x})=\left(\frac{4E\delta_{\max}}{(1-\mu^2)\alpha D_e^2}\left[(h-\frac{\delta}{2})((h-\delta)t+t^3)\right]\right)_{\delta=\delta_{\max}}-P_{\max}\ge 0,$$
$$g_3(\vec{x})=\delta_1-\delta_{\max}\ge 0,$$
$$g_4(\vec{x})=H-h-\mathrm{t}\ge 0,$$
$$g_5(\vec{x})=D_{\max}-D_e\ge 0,$$
$$g_6(\vec{x})=D_e-D_i\ge 0,$$
$$g_7(\vec{x})=0.3-\frac{h}{D_e-D_i}\ge 0.$$

where

$$\alpha=\frac{6}{\pi\ln K}\left(\frac{K-1}{K}\right)^2,$$
$$\beta=\frac{6}{\pi\ln K}\left(\frac{K-1}{\ln K}-1\right),$$
$$\gamma=\frac{6}{\pi\ln K}\left(\frac{K-1}{2}\right),$$

TABLE 23. Composition test functions.

Name	Function	D	Range	f_{opt}
$f_{24}(x)$ (CF1)	$f_1, f_2, f_3,\ldots, f_{10}$=Sphere Function $[\sigma_1, \sigma_2, \sigma_3,\ldots, \sigma_{10}]$=[1,1,1,…,1] $[\lambda_1, \lambda_2, \lambda_3,\ldots, \lambda_{10}]$=[5/100,5/100,5/100,…,5/100]	10	$[-5,5]^D$	0
$f_{25}(x)$ (CF2)	$f_1, f_2, f_3,\ldots, f_{10}$= Griewank0s Function $[\sigma_1, \sigma_2, \sigma_3,\ldots, \sigma_{10}]$=[1,1,1,…,1] $[\lambda_1, \lambda_2, \lambda_3,\ldots, \lambda_{10}]$=[5/100,5/100,5/100,…,5/100]	10	$[-5,5]^D$	0
$f_{26}(x)$ (CF3)	$f_1, f_2, f_3,\ldots, f_{10}$= Griewank0s Function $[\sigma_1, \sigma_2, \sigma_3,\ldots, \sigma_{10}]$=[1,1,1,…,1] $[\lambda_1, \lambda_2, \lambda_3,\ldots, \lambda_{10}]$=[1,1,1,…,1]	10	$[-5,5]^D$	0
$f_{27}(x)$ (CF4)	f_1, f_2 = Ackley0s Function f_3, f_4 = Rastrigin0s Function f_5, f_6 = Weierstrass Function f_7, f_8 = Griewank0s Function f_9, f_{10} = Sphere Function $[\sigma_1, \sigma_2, \sigma_3,\ldots, \sigma_{10}]$=[1,1,1,…,1] $[\lambda_1, \lambda_2, \lambda_3,\ldots, \lambda_{10}]$= [5/32,5/32,1,1,5/0.5,5/0.5,5/100,5/100,5/100, 5/100]	10	$[-5,5]^D$	0
$f_{28}(x)$ (CF5)	f_1, f_2 = Rastrigin0s Function f_3, f_4 = Weierstrass Function f_5, f_6 = Griewank0s Function f_7, f_8 = Ackley0s Function f_9, f_{10} = Sphere Function $[\sigma_1, \sigma_2, \sigma_3,\ldots, \sigma_{10}]$=[1,1,1,…,1] $[\lambda_1, \lambda_2, \lambda_3,\ldots, \lambda_{10}]$= [1/5,1/5, 5/0.5,5/0.5,5/100,5/100,5/32,5/32,5/100, 5/100]	10	$[-5,5]^D$	0
$f_{29}(x)$ (CF6)	f_1, f_2 = Rastrigin0s Function f_3, f_4 = Weierstrass Function f_5, f_6 = Griewank0s Function f_7, f_8 = Ackley0s Function f_9, f_{10} = Sphere Function $[\sigma_1, \sigma_2, \sigma_3,\ldots, \sigma_{10}]$=[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1] $[\lambda_1, \lambda_2, \lambda_3,\ldots, \lambda_{10}]$= [0.1×1/5,0.2×1/5, 0.3×5/0.5,0.4×5/0.5,0. 5×5/100,0. 6×5/100,0. 7×	10	$[-5,5]^D$	0

TABLE 24. Variation of $f(a)$ with a.

a	≤1.4	1.5	1.6	1.7	1.8	1.9	2	2.1
$f(a)$	1	0.85	0.77	0.71	0.66	0.63	0.6	0.58
a	2.2	2.3	2.4	2.5	2.6	2.7	≥2.8	
$f(a)$	0.56	0.55	0.53	0.52	0.51	0.51	0.5	

$$P_{\max} = 5400\text{lb}, \quad \delta_{\max} = 0.2 \text{ in}, \ S = 200000\text{Psi},$$
$$E = 30 \times 10^6\text{psi}, \quad \mu = 0.3, \ H = 2 \text{ in},$$
$$D_{\max} = 12.01 \text{ in}, \quad K = D_e/D_i, \ \delta_1 = f\,(a)\,h,$$
$$a = h/t.$$

Values of $f(a)$ vary as shown in Table 24.

Variable range $0.01 \le t \le 6$, $0.05 \le h \le 0.5$, $5 \le D_i \le 15$, $5 \le D_o \le 15$.

F. HYDROSTATIC THRUST BEARING DESIGN

Consider variable $\vec{x} = [R, R_o, \mu, Q]$.

$$\text{Minimize } f_6(\vec{x}) = \frac{QP_o}{0.7} + E_f.$$

$$\text{Subject to } g_1(\vec{x}) = W - W_s \ge 0,$$
$$g_2(\vec{x}) = P_{\max} - P_o \ge 0,$$
$$g_3(\vec{x}) = \Delta T_{\max} - \Delta T \ge 0,$$
$$g_4(\vec{x}) = h - h_{\min} \ge 0,$$
$$g_5(\vec{x}) = R - R_o \ge 0,$$
$$g_6(\vec{x}) = 0.001 - \frac{\gamma}{gP_o}\left(\frac{Q}{2\pi Rh}\right) \ge 0,$$
$$g_7(\vec{x}) = 5000 - \frac{W}{\pi(R^2 - R_o^2)} \ge 0.$$

where

$$W = \frac{\pi P_o}{2}\frac{R^2 - R_o^2}{\ln(R/R_o)},$$
$$P_o = \frac{6\mu Q}{\pi h^3}\ln(R/R_o), \quad E_f = 9336Q\gamma C\Delta T$$
$$\Delta T = 2(10^P - 560),$$
$$P = \frac{\log_{10}\log_{10}(8.122 \times 10^6\mu + 0.8) - C_1}{n},$$
$$h = \left(\frac{2\pi N}{60}\right)^2\frac{2\pi\mu}{E_f}\left(\frac{R^4}{4} - \frac{R_o^4}{4}\right), \quad \gamma = 0.0307, \ C = 0.5,$$
$$n = -3.55, \ C_1 = 10.04, \ W_s = 101000,$$
$$P_{\max} = 1000, h_{\min} = 0.001, \Delta T_{\max} = 50,$$
$$g = 386.4, N = 750.$$

Variable range $1 \le R \le 16$, $1 \le R_o \le 16$,

$$10^{-6} \le \mu \le 16 \times 10^{-6}, \quad 1 \le Q \le 16.$$

REFERENCES

[1] A. A. Goldstein, "On steepest descent," *J. Soc. Ind. Appl. Math. A, Control*, vol. 3, no. 1, pp. 147–151, 1965.

[2] S. Weerakoon and T. G. I. Fernando, "A variant of Newton's method with accelerated third-order convergence," *Appl. Math. Lett.*, vol. 13, no. 8, pp. 87–93, 2000.
[3] L. Wang, W. Zhao, Y. Tian, and G. Pan, "A bare bones bacterial foraging optimization algorithm," *Cognit. Syst. Res.*, vol. 52, pp. 301–311, Dec. 2018.
[4] E. G. Talbi, *Metaheuristics: From Design to Implementation*, vol. 74. Hoboken, NJ, USA: Wiley, 2009.
[5] J. Maddala and R. Rengaswamy, "A Genetic Algorithm (GA) based rational approach for design of discrete microfluidic networks," *Comput. Aided Chem. Eng.*, vol. 30, pp. 507–511, Jun. 2012.
[6] H. E. A. Ibrahima, F. N. Hassanb, and A. O. Shomerc, "Optimal PID control of a brushless DC motor using PSO and BF techniques," *Ain Shams Eng. J.*, vol. 5, no. 2, pp. 391–398, 2014.
[7] I. Chana, "Bacterial foraging based hyper-heuristic for resource scheduling in grid computing," *Future Gener. Comput. Syst.*, vol. 29, no. 3, pp. 751–762, 2013.
[8] C. Lai, Q. Shao, X. Chen, Z. Wang, X. Zhou, B. Yang, and L. Zhang, "Flood risk zoning using a rule mining based on ant colony algorithm," *J. Hydrol.*, vol. 542, pp. 268–280, Nov. 2016.
[9] C.-L. Hung and Y.-H. Wu, "Parallel genetic-based algorithm on multiple embedded graphic processing units for brain magnetic resonance imaging segmentation," *Comput. Elect. Eng.*, vol. 61, pp. 373–383, Jul. 2017.
[10] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Inf. Sci.*, vol. 237, pp. 82–117, Jul. 2013.
[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[12] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer, "Evolution algorithms in combinatorial optimization," *Parallel Comput.*, vol. 7, no. 1, pp. 65–85, 1988.
[13] J. Krause, J. Cordeiro, R. S. Parpinelli, and H. S. Lopes, "A survey of swarm algorithms applied to discrete optimization problems," in *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*. Philadelphia, PA, USA: Elsevier, 2013, pp. 169–191.
[14] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *J. Simul.*, vol. 76, no. 2, pp. 60–68, Feb. 2001.
[15] O. Montiel, O. Castillo, P. Melin, A. R. Díaz, and R. Sepúlveda, "Human evolutionary model: A new approach to optimization," *Inf. Sci.*, vol. 177, pp. 2075–2098, May 2007.
[16] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Syst. Mag.*, vol. 22, no. 3, pp. 52–67, Mar. 2002.
[17] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: Univ. of Michigan Press, 1975.
[18] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.
[19] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies—A comprehensive introduction," *Natural Comput.*, vol. 1, no. 1, pp. 3–52, 2002.
[20] P. Vikhar, "Evolutionary algorithm: A classical search and optimization technique," *Int. J. Pure Appl. Res. Eng. Technol.*, vol. 4, no. 9, pp. 758–766, 2016.
[21] F. Corno, M. S. Reorda, and G. Squillero, "A new evolutionary algorithm inspired by the selfish gene theory," in *Proc. IEEE Int. Conf. Evol. Comput., IEEE World Congr. Comput. Intell.*, May 1976, pp. 575–580.
[22] M. M. Eusuff and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *J. Water Resour. Planning Manage.*, vol. 129, no. 3, pp. 210–225, 2003.
[23] D. Simon, "Biogeography-based optimization," *IEEE Trans. Evol. Comput.*, vol. 12, no. 6, pp. 702–713, Dec. 2008.
[24] W.-T. Pan, "A new fruit fly optimization algorithm: Taking the financial distress model as an example," *Knowl.-Based Syst.*, vol. 26, pp. 69–74, Feb. 2012.
[25] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *J. Inf. Sci.*, vol. 179, no. 13, pp. 2232–2248, 2009.
[26] I. Bala and A. Yadav, "Gravitational search algorithm: A state-of-the-art review," in *Harmony Search and Nature Inspired Optimization Algorithms* (Advances in Intelligent Systems and Computing), vol. 741. Singapore: Springer, 2018, pp. 27–37.
[27] H. M. Genç, I. Eksin, and O. K. Erol, "Big Bang—Big Crunch optimization algorithm hybridized with local directional moves and application to target motion analysis problem," in *Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC)*, Oct. 2010, pp. 881–887.
[28] A. Kaveh and V. R. Mahdavi, "Colliding Bodies Optimization method for optimum discrete design of truss structures," *Comput. Struct.*, vol. 139, pp. 43–53, Jul. 2014.
[29] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: Charged system search," *Acta Mechanica*, vol. 213, no. 3, pp. 267–289, 2010.
[30] Z. Bayraktar, M. Komurcu, J. A. Bossard, and D. H. Werner, "The wind driven optimization technique and its application in electromagnetics," *IEEE Trans. Antennas Propag.*, vol. 61, no. 5, pp. 2745–2757, May 2013.
[31] R. A. Formato, "Central force optimization: A new metaheuristic with applications in applied electromagnetics," *Prog. Electromagn. Res.*, vol. 77, pp. 425–491, Dec. 2007.
[32] A. Kaveh and T. Bakhshpoori, "Water evaporation optimization: A novel physically inspired optimization algorithm," *Comput. Struct.*, vol. 167, pp. 69–85, Apr. 2016.
[33] H. Shah-Osseini, "Principal components analysis by the galaxy-based search algorithm: A novel metaheuristic for continuous optimisation," *Int. J. Comput. Sci. Eng.*, vol. 6, nos. 1–2, pp. 132–140, 2011.
[34] H. Abedinpourshotorban, S. M. Shamsuddin, Z. Beheshti, and D. N. A. Jawawi, "Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm," *Swarm Evol. Comput.*, vol. 26 pp. 8–22, Feb. 2016.
[35] W. Zhao, L. Wang, and Z. Zhang, "A novel atom search optimization for dispersion coefficient estimation in groundwater," *Future Gener. Comput. Syst.*, vol. 91, pp. 601–610, Feb. 2019.
[36] L. Xie and J. Zeng, "The performance analysis of artificial physics optimization algorithm driven by different virtual forces," *ICIC Exp. Lett.*, vol. 4, no. 1, pp. 239–244, 2010.
[37] A. Kaveh and A. Dadras, "A novel meta-heuristic optimization algorithm: Thermal exchange optimization," *Adv. Eng. Softw.*, vol. 110, pp. 69–84, Aug. 2017.
[38] D. P. Tripathi and U. R. Jena, "Cognitive and social information based PSO," *Int. J. Eng. Sci. Techno.*, vol. 8, no. 3, pp. 64–75, 2016.
[39] R. Oftadeh, M. J. Mahjoob, and M. Shariatpanahi, "A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search," *Comput. Math. Appl.*, vol. 60, no. 7, pp. 2087–2098, Oct. 2010.
[40] M. S. Kiran, "TSA: Tree-seed algorithm for continuous optimization," *Expert Syst. Appl.*, vol. 42, no. 19, pp. 6686–6698, 2015.
[41] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.
[42] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights, Nature & Biologically Inspired Computing," in *Proc. NaBIC World Congr.*, 2009, pp. 210–214.
[43] A. M. Fathollahi-Fard, M. Hajiaghaei-Keshteli, and R. Tavakkoli-Moghaddam, "The Social Engineering Optimizer (SEO)," *Eng. Appl. Artif. Intell.*, vol. 72, pp. 267–293, Jun. 2018.
[44] A. Kaveh and N. Farhoudi, "A new optimization method: Dolphin echolocation," *Adv. Eng. Softw.*, vol. 59, no. 5, pp. 53–70, May 2013.
[45] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *Int. J. Bio-Inspired Comput.*, vol. 2, no. 2, pp. 78–84, 2010.
[46] M. D. Li, H. Zhao, X. W. Weng, and T. Han, "A novel nature-inspired algorithm for optimization: Virus colony search," *Adv. Eng. Softw.*, vol. 92, pp. 65–88, Feb. 2016.
[47] A. Askarzadeh, "Bird mating optimizer: An optimization algorithm inspired by bird mating strategies," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 19, no. 4, pp. 1213–1228, 2014.
[48] G. Dhiman and V. Kumar, "Emperor penguin optimizer: A bio-inspired algorithm for engineering problems," *Knowl. Based Syst.*, vol. 159, pp. 20–50, Nov. 2018.
[49] A. H. Gandomi and A. H. Alavi, "Krill herd: A new bio-inspired optimization algorithm," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 17, pp. 4831–4845, May 2012.
[50] T. Ray and K. M. Liew, "Society and civilization: An optimization algorithm based on the simulation of social behavior," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 386–396, Aug. 2003.
[51] A. H. Kashan, "League championship algorithm: A new algorithm for numerical function optimization," in *Proc. Int. Conf. Soft Comput. Pattern Recognit. (SOCPAR)*, Singapore, Dec. 2009, pp. 43–48.
[52] S. Satapathy and A. Naik, "Social group optimization (SGO): A new population evolutionary optimization technique," *Complex Intel. Syst.*, vol. 2, no. 3, pp. 173–203, 2016.

[53] Y. Xu, Z. Cui, and J. Zeng, "Social emotional optimization algorithm for nonlinear constrained optimization problems," in *Swarm, Evolutionary, and Memetic Computing—SEMCCO* (Lecture Notes in Computer Science), vol. 6466. Berlin, Germany: Springer, 2010, pp. 583–590.

[54] M. Kumar, A. J. Kulkarni, and S. C. Satapathy, "Socio evolution & learning optimization algorithm: A socio-inspired optimization methodology," *Future Gener. Comput. Syst.*, vol. 81, pp. 252–272, Apr. 2018.

[55] T. T. Huan, A. J. Kulkarni, J. Kanesan, C. J. Huang, and A. Abraham, "Ideology algorithm: A socio-inspired optimization methodology," *Neural Comput. Appl.*, vol. 28, pp. 1–32, Dec. 2016.

[56] H. C. Kuo and C. H. Lin, "Cultural evolution algorithm for global optimizations and its applications," *J. Appl. Res. Technol.*, vol. 11, no. 4, pp. 510–522, 2013.

[57] N. S. Jaddi and S. Abdullah, "Optimization of neural network using kidney-inspired algorithm with control of filtration rate and chaotic map for real-world rainfall forecasting," *Eng. Appl. Artif. Intell.*, vol. 67, pp. 246–259, Jan. 2018.

[58] E. Alba and B. Dorronsoro, "The exploration/exploitation tradeoff in dynamic cellular genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 9, no. 2, pp. 126–142, Apr. 2005.

[59] D. H. Wolper and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.

[60] M. Ezekiel, "The cobweb theorem," *Quart. J. Econ.*, vol. 52, no. 2, pp. 255–280, 1938.

[61] F. V. Waugh, "Cobweb models," *Amer. J. Agricult. Econ.*, vol. 46, no. 4, pp. 732–750, 1964.

[62] W. Zhao and L. Wang, "An effective bacterial foraging optimizer for global optimization," *Inf. Sci.*, vol. 329, pp. 719–735, Feb. 2016.

[63] J. G. Digalakis and K. G. Margaritis, "On benchmarking functions for genetic algorithms," *Int. J. Comput. Math.*, vol. 77, no. 4, pp. 481–506, Sep. 2001.

[64] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, Tech. Rep. 201311, 2013.

[65] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Proc. IEEE Swarm Intell. Symp. (SIS)*, Jun. 2005, pp. 68–75.

[66] H. Nowcki, "Optimization in pre-contract ship design," in *Computer Applications in the Automation of Shipyard Operation and Ship Design*, vol. 2, Y. Fujita, K. Lind, and T. J. Williams, Eds. New York, NY, USA: Elsevier, 1974, pp. 327–338.

[67] H. Liu, Z. Cai, and Y. Wang, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," *Appl. Soft Comput.*, vol. 10, no. 2, pp. 629–640, Mar. 2010.

[68] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Inf. Sci.*, vol. 178, no. 15, pp. 3043–3074, Aug. 2008.

[69] Y. Wang, Z. Cai, Y. Zhou, and Z. Fan, "Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique," *Struct. Multidisciplinary Optim.*, vol. 37, no. 4, pp. 395–413, 2009.

[70] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems," *Eng. Comput.*, vol. 29, no. 1, pp. 17–35, 2011.

[71] H. Chickermane and H. C. Gea, "Structural optimization using a new local approximation method," *Int. J. Numer. Methods Eng.*, vol. 39, no. 5, pp. 829–846, 1996.

[72] M.-Y. Cheng and D. Prayogo, "Symbiotic organisms search: A new metaheuristic optimization algorithm," *Comput. Struct.*, vol. 139, pp. 98–112, Jul. 2014.

[73] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.

[74] A. D. Belegundu, "A study of mathematical programming methods for structural optimization," Ph.D. dissertation, Dept. Civil Environ. Eng., Univ. Iowa, Iowa City, IA, USA, 1982.

[75] C. A. C. Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Comput. Ind.*, vol. 41, no. 2, pp. 113–127, Mar. 2000.

[76] C. A. C. Coello and E. M. Montes, "Constraint-handling in genetic algorithms through the use of dominance-based tournament selection," *Adv. Eng. Inform.*, vol. 16, pp. 193–203, Jul. 2002.

[77] C. A. C. Coello and R. L. Becerra, "Efficient evolutionary optimization through the use of a cultural algorithm," *Eng. Optim.*, vol. 36, no. 2, pp. 219–236, 2004.

[78] Q. He and L. Wang, "An effective co-evolutionary particle swarm optimization for constrained engineering design problems," *Eng. Appl. Artif. Intell.*, vol. 20, no. 1, pp. 89–99, 2006.

[79] Q. He and L. Wang, "A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization," *Appl. Math. Comput.*, vol. 186, pp. 1407–1422, Mar. 2007.

[80] L. dos Santos Coelho, "Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems," *Expert Syst. Appl.*, vol. 37, no. 2, pp. 1676–1683, 2010.

[81] K. E. Parsopoulos and M. N. Vrahatis, "Unified particle swarm optimization for solving constrained engineering optimization problems," in *Proc. Int. Conf. Natural Comput.* Berlin, Germany: Springer, 2005, pp. 582–591.

[82] F.-Z. Huang, L. Wang, and Q. He, "An effective co-evolutionary differential evolution for constrained optimization," *Appl. Math. Comput.*, vol. 186, no. 1, pp. 340–356, 2007.

[83] E. Mezura-Montes and C. A. C. Coello, "Useful infeasible solutions in engineering optimization with evolutionary algorithms," in *Advances in Artificial Intelligence* (Lecture Notes in Computer Science), vol. 3789. Berlin, Germany: Springer, 2005, pp. 652–662.

[84] B. R. Rao and R. Tiwari, "Optimum design of rolling element bearings using genetic algorithms," *Mechanism Mach. Theory*, vol. 42, no. 2, pp. 233–250, 2007.

[85] S. Gupta, R. Tiwari, and B. N. Shivashankar, "Multi-objective design optimisation of rolling bearings using genetic algorithms," *Mechanism Mach. Theory*, vol. 42, pp. 1418–1443, Oct. 2017.

[86] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems," *Comput.-Aided Des.*, vol. 43, no. 3, pp. 303–315, 2011.

[87] A. Sadollah, A. Bahreininejad, H. Eskandar, and M. Hamdi, "Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2592–2612, May 2013.

[88] C. A. C. Coello, "Treating constraints as objectives for single-objective evolutionary optimization," *Eng. Opt.*, vol. 35, vol. 32, no. 3, pp. 275–308, 2000.

[89] K. Deb and M. Goyal, "Optimizing engineering designs using a combined genetic search," in *Proc. 7th Int. Conf. Genetic Algorithms*, I. T. Back, Ed., 1997, pp. 512–528.

[90] J. N. Siddall, *Optimal Engineering Design: Principles and Applications*. New York, NY, USA: Marcel Dekker, 1982.

[91] S. He, E. Prempain, and Q. H. Wu, "An improved particle swarm optimizer for mechanical design optimization problems," *Eng. Optim.*, vol. 36, no. 5, pp. 585–605, Oct. 2004.

WEIGUO ZHAO received the Ph.D. degree from the School of Electrical Engineering, Hebei University of Technology, in 2016. He is currently an Associate Professor with the Hebei University of Engineering and also a Scholar with the University of Illinois at Urbana–Champaign. He has published over 30 research papers in international journals. He is currently involved in the areas of soft computing, intelligent fault diagnosis, and hydrologic modeling.

LIYING WANG received the Ph.D. degree from Beijng Jiaotong University, China, in 2014. She is currently a Professor with the Hebei University of Engineering. She was a Postdoctoral Researcher with the Institute of Technology, University of Ontario, in 2015, and the Hebei University of Engineering, in 2017. She has authored over 50 scientific papers. She has authored and published one monograph. Her current research interests include intelligent computing, control systems engineering, and hydraulic engineering. She has received one hundred outstanding innovative scholars of colleges and universities in Hebei Province, in 2019.

ZHENXING ZHANG received the B.Sc. degree in environment science from Wuhan University, the M.Sc. degree in environment science from Peking University, the M.Sc. degree in applied statistics from the University of Syracuse, and the Ph.D. degree in water resources engineering from the State University of New York College of Environmental Science and Forestry (SUNY ESF). He is a Hydrologist with the Prairie Research Institute, University of Illinois at Urbana-Champaign, focusing on integrate water management, surface water supply, water availability, hydrologic modeling, and stochastic hydrology. He is a licensed Professional Engineer (P.E). He has published over 30 papers in top journals including *Water Resources Research*, the *Journal of Hydrology*, *Applied Energy*, and *Desalination*.