Applied Soft Computing 64 (2018) 161–185

Contents lists available at ScienceDirect

Applied Soft Computing

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

Volleyball Premier League Algorithm

Reza Moghdani, Khodakaram Salimifard*

Industrial Management Department, Persian Gulf University, Bushehr, Iran

ARTICLE INFO

Article history:
Received 16 February 2017
Received in revised form
23 November 2017
Accepted 27 November 2017
Available online 2 December 2017

Keywords:
Metaheuristic
Optimization
Global optimization
Volleyball premier league

ABSTRACT

This article proposes a novel metaheuristic algorithm called Volleyball Premier League (VPL) inspired by the competition and interaction among volleyball teams during a season. It also mimics the coaching process during a volleyball match. To solve global optimization problems using the volleyball metaphor, there are terms such as substitution, coaching, and learning, which are captured in the VPL algorithm. The proposed algorithm is benchmarked on 23 well-known test functions, which are categorized into three groups, namely unimodal, multimodal and fixed-dimension multimodal functions. The solutions obtained using the VPL have been compared with other metaheuristic algorithms including Particle Swarm Optimization (PSO), Differential Evolution (DE), Genetic Algorithm (GA), Artificial Bee Colony (ABC), Firefly Algorithm (FA), Harmony Search (HS), Sin Cosine Algorithm (SCA), Soccer League Competition (SLC), and League Championship Algorithm (LCA). In addition, VPL has been used to solve three classical engineering design optimization problems. Results show that VPL algorithm possesses a strong capability to produce superior performance over the other well-known metaheuristic algorithms. The results of the experiments also show that the VPL is effectively applicable to solve problems with complex search space.

1. Introduction

In order to find effective solution methods to optimization problems, many researchers have tried to propose new algorithms and/or improve the existing methods. There are many optimization problems, which are extremely difficult to solve, and scholars and researchers have investigated new methods to deal with them. To cope with these problems, useful strategies have been implemented in numerical optimization algorithms to obtain the global optimum.

If an optimization problem has an unknown search space, the resulting solution may depend on the selection of an initial point [1]. In these kinds of problems, search space grows rapidly while the dimensions of the problem are increased. Hence, it is widely debated that classical optimization methods do not determine an optimum solution in reasonable time. Mirjalili [2] discussed the reasons why metaheuristic algorithms have become notably common. Plainness, flexibility, derivative-free mechanism, and local optima avoidance are the four main reasons, which motivate scholars to use metaheuristic algorithms. There are many metaheuristic algorithms proposed to solve optimization problems. A wide range of complex problems such as scheduling [3–6], data clustering [7–9], image and video processing [10–12], tuning of neural networks [13–15], bin packing [16–18], portfolio selection [19–21], and pattern recognition [22,23] have been efficiently solved using metaheuristic algorithms.

The growth in the number of evolutionary optimization techniques and metaheuristic algorithms have been impressive. Metaheuristic algorithms combine some deterministic and random behavior to emulate natural phenomena [24]. One of the most common evolutionary optimization techniques is the genetic algorithm (GA) proposed by Holland [25] and further developed by Goldberg [26]. There are many other evolutionary optimization methods proposed in literature [2,27–30]. Metaheuristic algorithms have been classified in different ways [29,31,32]. The main branch of metaheuristic algorithms can be classified into four categories including stochastic, evolutionary, physical, and swarm algorithms.

Stochastic optimization algorithms use randomness to perform non-deterministic behaviors, generally different from purely deterministic procedures. The well-known algorithms in this category are local search (LS) [33], random search [34], adaptive random search (ARS) [35], stochastic hill climbing (SHC) [36], hill climbing (HC) [37], iterated local search (ILS) [38], guided local search (GLS) [33], variable neighborhood search (VNS) [39], greedy randomized adaptive search procedure (GRASP) [39], tabu search (TS)

* Corresponding author.
E-mail addresses: reza.moghdani@gmail.com (R. Moghdani), salimifard@pgu.ac.ir (K. Salimifard).

https://doi.org/10.1016/j.asoc.2017.11.043
1568-4946/

[40], reactive tabu search (RTS) [41], and vortex search algorithm (VSA) [42].

Recently, researchers try to embed local search mechanism into the framework of population-based evolutionary algorithms [43]. Evolutionary algorithms concerned with computational methods belong to the evolutionary computation field of study [44]. The most famous algorithm in this category is Genetic Algorithm, which is known as the basis of many other metaheuristic algorithms. The other algorithms in this category are evolution strategies (ES) [45], covariance matrix adaptation evolution strategies (CMA-ES) [46], evolutionary programming (EP) [47], differential evolution (DE) [48], fast evolutionary programming (FEP) [49], grammatical evolution (GE) [50], gene expression programming (GEP) [51], adaptive dimensional search (ADS) [52], adaptive differential evolution (ADE) [53], interior search algorithm (ISA) [54], opposition-based algorithm (OBA) [55], and stochastic fractal search (SFS) [56].

Physical algorithms are those inspired by a physical process [44]. The inspiring physical systems range from metallurgy, music, the interplay between culture and evolution, science (chemistry, physics, mathematics), and complex dynamic systems. In general, a physical algorithm is a combination of local (neighborhood-based) and global search techniques. The most famous algorithm in this category is simulated annealing (SA). This algorithm is also called the threshold algorithm which belongs to the category of local search algorithms [57]. It is believed that SA is generally adapted from the Metropolis-Hastings Monte Carlo algorithm. Similar to the GA, a large variety of extensions and specialization of methods, not limited to parallel simulated annealing, fast simulated annealing, and adaptive simulated annealing, is branched from the SA which are extensively used in different optimization problems. The other algorithms of this category include fields of forces (FOF) [58], charged system search (CSS) [59], ray optimization algorithm (ROA) [60], optics inspired optimization (OIO) [61], galaxy-based search algorithm (GbSA) [62], electromagnetism algorithm (EMA) [63], BB–BC [64], colliding bodies optimization (CBO) [65], and harmony search (HS) [66]. This category also includes cultural algorithm (CA) [67], imperialist competitive algorithm (ICA) [68], biogeography-based optimization (BBO) [69], ecogeography-based optimization (EBO) [70], chemical reaction optimization (CRO) [71], ions motion algorithm (IMA) [28], and bacteria chemotaxis (BC) [72]. Water cycle algorithm (WCA) [73], colonizing weeds (CW) [74], flower pollination algorithm (FPA) [75], intelligent water drops (IWD) [76], runner-root algorithm (RRA) [77], forest optimization algorithm (FOA) [78], tree seed algorithm (TSA) [79], water wave optimization (WWO) [80], teaching–learning-based optimization (TLBO) [81], lightning search algorithm (LSA) [82], mine blast algorithm (MBA) [1], and grenade explosion method (GEM) [83] are also classified in this category.

In the last category, swarm-based algorithms imitate the social and individual behavior of swarms, animals, herds, teams, or any group of creatures. The most well-known algorithm in the area of swarm intelligence is particle swarm optimization (PSO), which is based on the behavior of the flock of birds [84]. From the viewpoint of initializing with a population of random solutions, PSO is similar to GA. Another well-known algorithm in this category is ant colony optimization (ACO) introduced by Dorigo [85–87]. It is worth mentioning here that there are many swarm based metaheuristic methods in the literature such as cuckoo search algorithm (CS) [88], migrating birds optimization (MBO) [89], grey wolf optimizer (GWO) [2], bees algorithm (BA) [90], artificial fish swarm algorithm (AFS) [91], symbiotic organisms search (SOS) [92], spider optimization algorithm (SOA) [93], and shuffled frog leaping (SFL) [94]. Monkey search algorithm (MSA) [95] Keshtel algorithm (KA) [96], cat swarm optimization (CSO) [97], dolphin echolocation (DE) [98], krill herd algorithm (KH) [99], moth flame optimization (MFO) [100], elephant herding optimization (EHO) [101], dragonfly algorithm (DA) [102], firefly algorithm (FA) [103], glowworm swarm optimization (GSO) [104], heart algorithm (HA) [105], artificial bee colony (ABC) [106], and lion optimization algorithm (LOA) [107] are also classified in this category.

Based on group behavior and some artificial physical processes, in this paper, an optimization algorithm namely Volleyball Premier League (VPL) is introduced. Moosavian [108] and Kashan [109] proposed two algorithms inspired by characteristics of sport. We have been inspired by the general ideas and rules defined in [108,109]. We have also specified some new metaphors and terms, which are related to volleyball. Our novel algorithm is inspired, in its core concept, by simulation of activities in a volleyball game. The proposed algorithm is truly different from the other algorithms in the literature.

The remaining structure of this article is organized as follows. Section 2 introduces the Volleyball Premier League (VPL) algorithm. Section 3 covers experiments and results of comparing VPL with other well-known algorithms. Section 4 is devoted to experiments with test functions and applications from engineering design problems. Finally, the article is summarized and concluded in Section 5.

2. The VPL algorithm

The proposed VPL algorithm mimics the interactive behavior among teams in a volleyball league. It also captures the coaching decisions during a match. The main inspiration of the proposed algorithm is expressed first, and then, the mathematical model is presented.

Over one hundred years ago, in 1895, William G. Morgan introduced a new game called *Mintonette.* In 1896, this game changed its name to volleyball because of the volleying nature of the game [110]. In a match, a net separates two teams of six players. In order to gain a point, each team must ground the ball on the other team's court under some regulated rules. The match progressed as follows: a player on one side starts a rally by trying to serve the ball over the net and other team's court. The rally will continue until a team performs a fault, and then a point will be awarded to another team.

The head coach of a team submits the team's starting lineup to the second referee. To start a rally, all players are set on the court. Typically, three players are assigned to the front-row and the others are placed in the back-row. The typical position of members of teams in the court is depicted in Fig. 1.

Each team can form its own special lineup according to its players. The coach usually stands near the court to lead players during the match. Substitutes sit on the bench while they are informed and trained by the coach and analysts. They are also ready to play in the match if needed. As it is shown in Fig. 1, there are three different interactions between team members and the coach. In order to capture these interactions, novel operators are implemented in the proposed algorithm.

2.1. Solution representation

As it was mentioned before, team members are grouped as players, substitutes, and the coach. The special feature of the proposed algorithm is the solution representation structure, which is generally different from any other metaheuristic algorithms. The solution representation consists of two segments named the *active* and the *passive* parts. The active part represents the main formulation of each team, which includes six active players. The fitness function for each solution is computed according to this part. The passive part holds some variables, to present special inspiration rules, like substitution strategy. In reality, a substitute can occupy the position of the player who is leaving the court according to the head

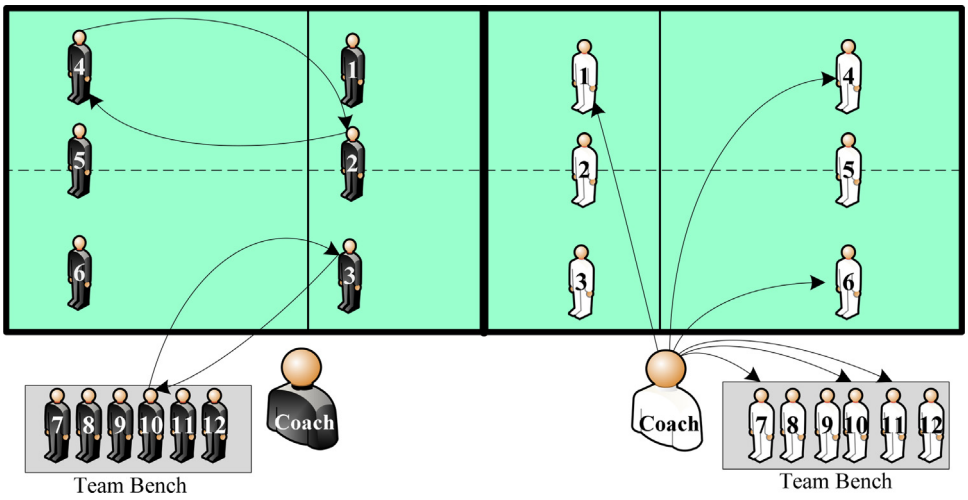

Fig. 1. Typical court of the volleyball game and interactions involved.

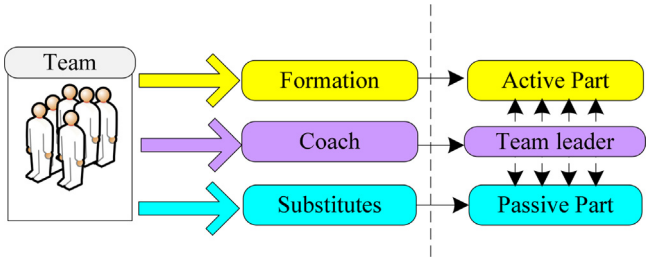

Fig. 2. Relationship between team structure and the solution representation.

coach's decision. This action is called substitution that is authorized by referees.

Fig. 2 shows the relationship between the special structure of the team and the solution representation of the algorithm. As it can be seen, the active part is defined based on the team formation, while the passive part of the solution is determined based on the substitutes. The team leader (coach) decisions can manipulate, directly or indirectly, both active and passive parts of the solution representation.

A team formation determines the general style of the team on the court. Generally, three standard formations are used in volleyball, namely the "4-2", "2-4" and "5-1" formations. A formation "H-S" refers to the number of hitters (H) and setters (S) players, respectively. The coach decides the formation in accordance with the team circumstances. Rationally, different formations can be used depending on whether a team wants to play more offensive or defensive. Therefore, a coach must find the best formation, which is the outcome of the type and the quality of available active players and substitutes. Hence, the formation and substitutes may be changed during a match, based on the coach approach and knowledge. In VPL, it is assumed that a team has the same number of active players and substitutes. The structure of the solution representation is shown in Fig. 3.

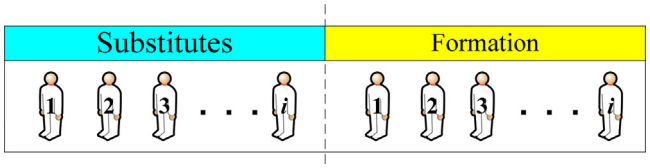

Fig. 3. Solution representation.

In a regular volleyball league, many factors affect the match results. To reach a general idea to implement the algorithm, the following assumptions are made in the algorithm design process:

- The result of each match is not foreknown. It means that any result can be seen at the end of a match. Therefore, it is possible that one of the weakest teams beats the top team in the league.
- The term "team power" is defined to emphasize the fact that it is more likely that the stronger team defeats its opponent.
- A team pays attention to its forthcoming game and does not consider any other future games. In respect to the previous results, the new lineup of each team in the court is based on the analysis of the current team power and the chance to win the match.
- When team i defeats team j, any strength that helped team i to win will be considered as the weakness of team j. This means a weakness is defined as the lack of a special strength.

The main steps of the Volleyball Premier League (VPL) are shown in Fig. 4.

We use the term "league" to represent the population concept. In addition, the term "number of seasons" is used as the current iteration number. A team represents a solution, such that team i represents the ith member of the population (league). The term "week" is used to represent league schedule, which will be explained in the next sub-section.

2.2. Initialization

Similar to the other optimization algorithms, VPL algorithm starts with the initialization of the teams which represents the set of initial solutions to the problem. As we mentioned above, each team has two main properties, formation and substitutes. The initial set of teams is defined by $TEAM^0$ with the population of NT teams, where NT denotes the number of teams (population size) in each season. Initially, let X_j^f and X_j^s be the formation and the substitute properties of the jth variable which are assigned random numbers between lower and upper bounds of each variable using Eqs. (1) and (2).

$$X_j^f = lb_j + Rand() \times \left(ub_j - lb_j\right) \quad (1)$$

$$X_j^s = lb_j + Rand() \times \left(ub_j - lb_j\right) \quad (2)$$

lb_j and ub_j are lower bound and upper bound of the variable j, respectively. $Rand()$ is a uniformly distributed random number between 0 and 1. The main properties of initial solutions are

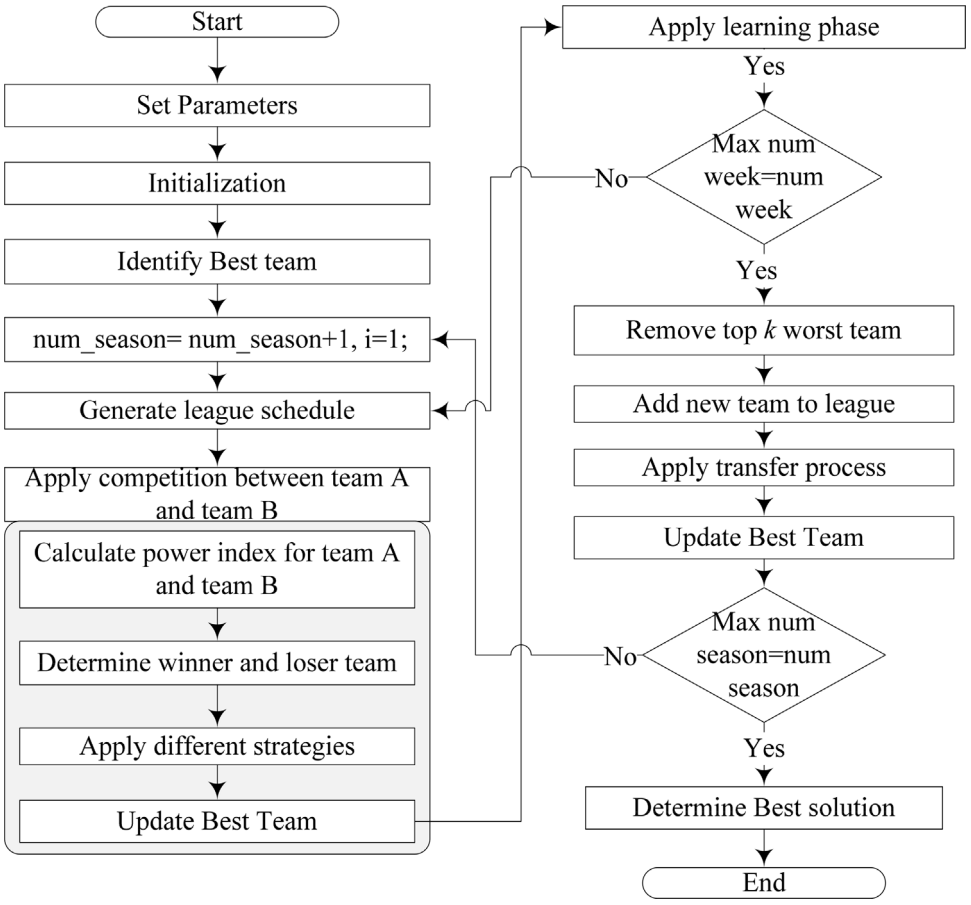

Fig. 4. Flowchart of the volleyball premier league (VPL) algorithm.

demonstrated with matrices F and S, where the number of rows and columns indicate the number of teams (NT) and the number of dimensions, respectively. The value for NT can be set wisely, while the number of dimensions is identical to the number of variables. The matrices representing formation and substitutes properties of teams are defined according to Eqs. (3) and (4), respectively.

$$F = \begin{bmatrix} X^f_{1,1} & X^f_{1,2} & \cdots & X^f_{1,j} \\ X^f_{2,1} & X^f_{2,2} & \cdots & X^f_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ X^f_{i,1} & X^f_{i,2} & \cdots & X^f_{i,j} \end{bmatrix} \tag{3}$$

$$S = \begin{bmatrix} X^s_{1,1} & X^s_{1,2} & \cdots & X^s_{1,j} \\ X^s_{2,1} & X^s_{2,2} & \cdots & X^s_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ X^s_{i,1} & X^s_{i,2} & \cdots & X^s_{i,j} \end{bmatrix} \tag{4}$$

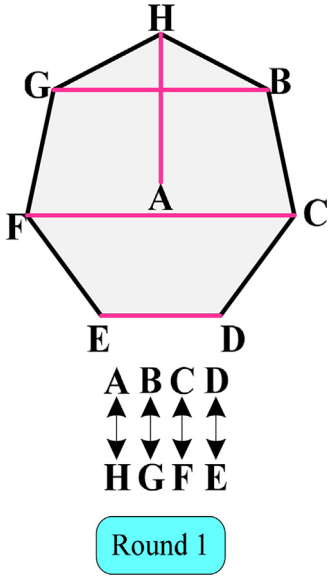

Fig. 5. First round of SSR method.

2.3. Match schedule

In a typical sports tournament, there is a need for a games schedule. In recent years, sports scheduling has become a research area in operations research and computer science communities [111]. In order to have an optimized solution, it is required for a metaheuristic algorithm to generate game schedules. We use the single round robin (SRR) method to generate the league's schedule, artificially. In this step, each team faces the other teams exactly once in a single round. There is a reiteration of action whenever the same pair of teams face each other twice in two consecutive rounds. If the number of teams is not even, then in each round, one team does not play. In order to have an even number of teams, we can add a dummy team.

In an SRR tournament, a team plays against all the other teams. There is a systematic approach to schedule a round robin tour-

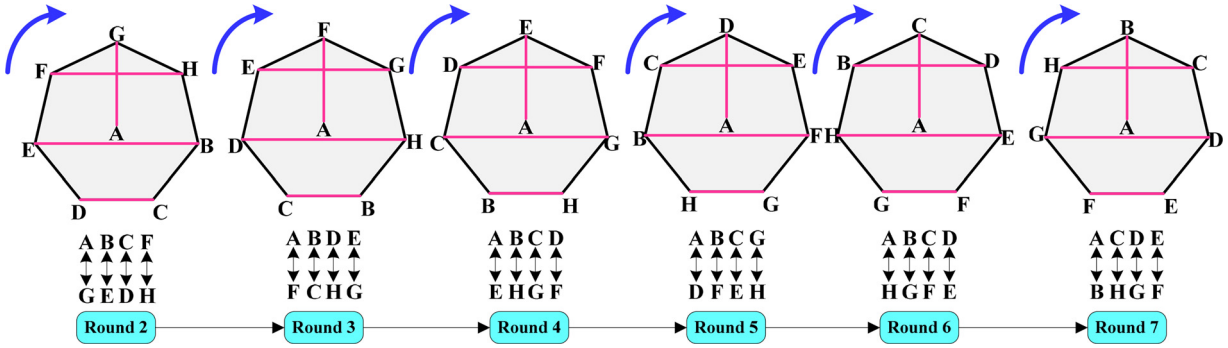

Fig. 6. Generating league schedule for rounds 2–7.

Table 1
A typical league schedule.

Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
A-H	A-G	A-F	A-E	A-D	A-H	A-B
B-G	B-E	B-C	B-H	B-F	B-G	C-H
C-F	C-D	D-H	C-G	C-E	C-F	D-G
D-E	F-H	E-G	D-F	G-H	D-E	E-F

nament. We use a simple procedure named the polygon method. Let N be the number of teams in the tournament and each team plays $N-1$ games. Totally, $(N-1)N/2$ games will be played during a tournament. Suppose we have eight teams in the league. In order to run SRR, we draw a regular $N-1$ sided polygon, so that each vertex and the center point symbolize a team. As shown in Fig. 5, horizontal lines link two vertices on the opposite sides of the polygon. Another line is used to link the center point to the vertex on the top. Each line represents a match between the two teams represented by the vertices on the ending point of the line and they will be playing in the first round.

According to Fig. 5, $(A, H), (B, G), (C, F)$ and (C, F) play in the first round. In order to determine the league schedule in the next round, the polygon is rotated clockwise. The new pairings define round two. This procedure will continue until the polygon returns to its original position (Fig. 6).

As illustrated in Fig. 6, the typical league schedule for eight teams is shown in Table 1.

2.4. Competition

Regularly, each team plays one match a week. In a volleyball match, there is no draw and the results for a team is either win or lose. To determine the winning team, 3–5 sets are played. A set is played until a team gains 25 points, on the condition that there is at least a two-point margin of victory. The winning team is chosen by acquiring three-out-of-five sets in the match. If a fifth set is played, the team that scores 15 points first is the winner, in the similar condition of sets with 25 points.

In order to determine the winning team in a competition, we present mathematical equations expressing for a team the chance of winning and the power to win the match. We suppose that the relationship among power of each team resulted from linear equations. Here we suppose that competitions among teams are idealized, and there are no unpredictable factors that have an effect on the match result. In order to determine the strength of each team in a week and the formation of the team i defined as X_i^f, we introduce team power index as follows:

$$\varphi(i) = \frac{f(X_i^f)}{Z} \tag{5}$$

$$Z = \sum_{i=1}^{n} f(X_i^f) \tag{6}$$

Where $\varphi(i)$ indicates the power index for the team i, $f(X_i^f)$ the fitness value of team i is computed from its formation and Z indicates the total sum of fitness values in a week. Eq. (5) implies that the team power is proportional to the fitness value of all teams. The stronger the team, the higher the value of $\varphi(i)$. Since each team is measured according to its weight, the chance of winning a match can be calculated using the fitness value. Let us consider two teams j and k playing in a match, with their formations X_j^f and X_k^f, respectively. So, the power index for teams are computed as follows:

$$\varphi(j) = \frac{f(X_j^f)}{Z} \tag{7}$$

$$\varphi(k) = \frac{f(X_k^f)}{Z} \tag{8}$$

Let $p(j, k)$ indicates the probability that team j wins the match. Then we have

$$p(j, k) = \frac{\varphi(j)}{\varphi(j) + \varphi(k)} \tag{9}$$

In any match between two teams k and j, we have the following probability principle:

$$p(j, k) + p(k, j) = 1 \tag{10}$$

The following formula is also true:

$$p(k, j) = 1 - \frac{\varphi(j)}{\varphi(j) + \varphi(k)} \tag{11}$$

Since $p(j, k)$ shows the chance of winning a match, the winner of any match can be determined using a uniformly distributed random number $r \in [0, 1]$. If $r \leq p(j, k)$, team j wins the match, otherwise the team k is the winner. Clearly, if $f(X_j^f)$ and $f(X_k^f)$ are close to each other, $p(j, k)$ and $p(k, j)$ tend to 0.5. After determining the winner of the match, to set a new formation, strategies for winner and loser teams are applied. Knowledge sharing, repositioning, and substitution are the three strategies which can be considered for a losing team, whilst the winning team can use the leading role strategy.

Table 2
Pseudo-code of competition between team i and team j.

Function Competition (i, j)
  Calculate $\varphi(i)$ and Z using Eqs (5) and (6),
  Calculate $p(i,j)$ using Eq (9),
  Generate $r \in [0,1]$,
  If $p(i,j) \leq r$
    team i is the winner and team j is the loser,
  Else
    team j is the winner and team i is the loser,
  End if
  apply winning strategy for the winner,
  apply losing strategies for the loser,
End

According to the aforementioned explanation in analyzing each match, the procedure of competition is as shown in Table 2:

2.5. Knowledge sharing strategy

The volleyball coach has a remarkable effect on the team performance during a match. He is responsible for coaching players. He usually assesses players' skills during a match and assigns team positions. One of the most prominent tasks of a coach is to analyze the performance of his team members and the opposing players during a match. Based on the analysis, he makes an update on the teams technical and tactical knowledge related to the match. As shown in Fig. 1, a coach shares his knowledge with players and substitutes.

Coaches usually use a temporal analysis concerning the duration of the rallies, breaks and sets and train players and substitutes during the match. We named these activities as knowledge sharing which may help a losing team to perform better in the match. The mathematical formulations of this strategy are shown below:

$$X_j^f(t+1) = X_j^f(t) + r_1\lambda^f\left(ub_j - lb_j\right) \tag{12}$$

$$X_j^s(t+1) = X_j^s(t) + r_2\lambda^s\left(ub_j - lb_j\right) \tag{13}$$

Where λ^f and λ^s are coefficients of formation and substitutes, respectively. r_1 and r_2 are random numbers uniformly distributed between 0 and 1. δ_{ks} indicates the rate of knowledge sharing in each team. The following formulation calculates the number of knowledge sharing in each competition.

$$N_{ks} = \left[J\delta_{ks}\right] \tag{14}$$

Where N_{ks} denotes the number of knowledge sharing positions for each team and J represents the number of total positions in each team. Having all the notations and formulas of knowledge sharing strategy been defined, pseudo-code for knowledge sharing strategy is shown in Table 3.

Coaching for a game can be done in two ways: match coaching and developmental coaching. Obviously, the main goal of a coach is to win a match. Developmental coaching emphasizes on the players development through the reinforcement of basic skills during exercises known as drills. A coach designs drills that simulate match situations thereby encouraging speed of movement, anticipation, timing, communication, and team-work, aiming to meet the strategic requirements of the team.

Table 3
Pseudo-code for knowledge sharing strategy.

For k=1 : N_{ks}
  Select randomly a position
  For j=1 to J
    Update position j of formation property using Eq (12),
    Update position j of substitutes property using Eq (13),
  End For
End For

2.6. Repositioning strategy

In the volleyball game, each position determines what a player should to do on the court during a match. Players have a list of duties and they are going to make the highest possible of coordination to make the best result. Roles include middle blocker, outside hitter, libero, setter, and opposite. Coaches try to put the most suitable player in each role in order to gain the best performance in the league. Because of some difficulties, such as shortage of financial resources, teams may not afford to recruit all the desired players. So they have to use current players and attempt to enhance their performance during league matches. Players are trained under the coaching leadership so that they are skilled to play various roles. Hence, players can be assigned to different positions during a match. We call this procedure as repositioning strategy such that a team changes positions of active players to gain a better performance. A possible use of repositioning strategy is illustrated in Fig. 1 where the left team swaps the positions of the players 2 with 4. As it is shown, this strategy can be applied on substitution bench, as well.

For each losing team, we will perform this procedure. Let δ_{rs} indicates the rate of repositioning procedure in each team. Eq. (15) calculates the number of repositioning procedures in each competition.

$$N_{rs} = \left[J\delta_{rs}\right] \tag{15}$$

Where N_{rs} denotes the number of repositioning procedure in each competition, after selecting two positions i and j randomly, two variables A and B with two properties formation and substitutes are defined, and then we assign properties of positions i and j to properties of A and B, respectively. So we get the following formulas:

$$A^f = X_i^f \tag{16}$$

$$A^s = X_i^s \tag{17}$$

$$B^f = X_j^f \tag{18}$$

$$B^s = X_j^s \tag{19}$$

Then, we assign properties of variables A and B to properties of j and i, respectively. So we get the following formulas:

$$X_i^f = B^f \tag{20}$$

$$X_i^s = B^s \tag{21}$$

$$X_j^f = A^f \tag{22}$$

$$X_j^s = A^s \tag{23}$$

The pseudo-code of repositioning process is expressed in Table 4.

Table 4
Pseudo-code for repositioning process.

For k=1 to N_{rs}
  Select randomly two positions i and j
  Define two variables A and B
  Use formulas (16) -(19),
  Reverse two position i and j using (20)- (22)
End

Table 5
Pseudo-code for substitution process.

Compute number of substitution process using Eq (15)
Define sets h, F, and S
For k=1 to N_s
$X^f_{h(k)} = S(k)$
$X^s_{h(k)} = F(k)$
End

2.7. Substitution strategy

Substitution is a special process in volleyball games in which the head coach replaces players on the bench with ones on the court. Contrary to a game like football, there is no limitation on the number of substitution, so this process can be chosen after each rally. Coaches must analyze players who can play well and the combinations that play well together to maximize the team success. During the match, a substitute may be swapped by another substitute or by a player who is currently a member of substations. One of the most important duties of a coach is to choose a good substitution strategy which makes different formation by changing players' positions in the substitution process.

We will use substitution process to have a better search in the algorithm. This process is clearly shown for black team as shown in Fig. 1. In the original version of volleyball rules, there were no regulations concerning the number of players coming into the game to substitute for other players. Today, teams are limited in the number of substitutions they can make in a match, but we commit the algorithm to follow the original version of volleyball rules, so the number of substitutions is not limited. Let r indicates a uniformly distributed random number between 0 and 1. Eq. (24) calculates the number of substitution in each competition.

$$N_s = [rJ] \tag{24}$$

N_s denotes the number of substitution procedures in each team and J represents the number of positions. In each competition, after determining loser team (selected indices of positions randomly named set h), sets F and S are defined which contain players and substitutions candidates, and then all members of sets F and S are exchanged randomly. The pseudo-code for substitution process is presented in Table 5.

2.8. Winner strategy

The main idea of this part of the algorithm is to use computational intelligence inspired by social interaction. In VPL, a number of simple entities, the teams, are located in the search space of the problem, and each one is evaluated based on the objective function at its current location. Each winning team then detects its position within the search space by combining some features of the history of the best team and some random variations.

In this strategy, the new position of a team is defined based on itself $X^g(t)$, the best team $X^g(t)^*$, and the inertia weight ψ^g. Following formulas are defined for winning strategy.

$$X^f(t+1) = X^f(t) + r_1\psi^f\left(X^f(t)^* - X^f(t)\right) \tag{25}$$

$$X^s(t+1) = X^s(t) + r_2\psi^s\left(X^s(t)^* - X^s(t)\right) \tag{26}$$

Where ψ^f and ψ^s denote inertia weights of formation and substitutes, and r_1 and r_2 are two uniformly distributed random numbers between 0 and 1, respectively.

2.9. Learning phase

In order to capture the learning phase of teams, we consider the best solution as the *rank*1. Consequently, the second and the third best solutions are named *rank*2 and *rank*3, respectively. The remaining teams are assumed to follow these three teams. We define the following equations to capture the learning behavior of a team.

$$\theta = dbr_1 - b \tag{27}$$

$$\vartheta = dr_2 \tag{28}$$

where θ and ϑ are coefficient values, d is equal to constant β, r_1 and r_2 are two uniformly distributed random numbers between 0 and 1, and b is linearly decreased from β to 0. We define Eq. (29) to calculate the value of b.

$$b = \beta - \left(\mathrm{t}\left(\beta/T\right)\right) \tag{29}$$

Where t denotes the current iteration, and T represents the maximum number of iterations in the algorithm. Using θ and ϑ, the following formula is defined.

$$X^g_j(t+1)_\Phi = \left(X^g_j(t)\right)_\Phi - \theta\left(\left|\vartheta\left(X^g_j(t)\right)_\Phi - X^g_j(t)\right|\right) \tag{30}$$

In Eq. (30), we have six sets, generated by the sets $g = \{s, f\}$ and $\Phi = \{1, 2, 3\}$, where s and f denote the formation and substitutes properties of the team. The index Φ may take a value 1–3, representing *rank*1, *rank*2, and *rank*3 teams of the current iteration. $X^g_j(t)$ is the value of position j, and $X^g_j(t+1)_\Phi$ shows the value of the position j of property g related to the best solutions Φ.

A volleyball coach analyzes the performance of all teams in the premier league. Also, he may hire professional analysts to help him deeply understand what the leading teams in the league do to have an ideal performance. He usually coaches the players in a way to get closer to the performance of team *rank*1. Teams *rank*2 and *rank*3 are also considered. Therefore, the best formation and substitute properties of the best possible team are not known. For the purpose of mathematical modeling of the learning process, we assume that teams *rank*1, *rank*2, and *rank*3 have better knowledge about the best possible formation and substitutes. Following formulas are defined to capture the learning phase.

$$X^f_j(t+1)_1 = \left(X^f_j(t)\right)_1 - \theta\left(\left|\vartheta\left(X^f_j(t)\right)_1 - X^f_j(t)\right|\right) \tag{31}$$

$$X^f_j(t+1)_2 = \left(X^f_j(t)\right)_2 - \theta\left(\left|\vartheta\left(X^f_j(t)\right)_2 - X^f_j(t)\right|\right) \tag{32}$$

$$X^f_j(t+1)_3 = \left(X^f_j(t)\right)_3 - \theta\left(\left|\vartheta\left(X^f_j(t)\right)_2 - X^f_j(t)\right|\right) \tag{33}$$

We consider three best teams (*rank*1, *rank*2, and *rank*3), and coerce the current solution to update its properties according to the best teams properties. In this consideration, Eq. (34) is defined.

$$X^s_j(t+1)_1 = \left(X^s_j(t)\right)_1 - \theta\left(\left|\vartheta\left(X^s_j(t)\right)_1 - X^s_j(t)\right|\right) \tag{34}$$

Similar to formation property of the solution, following formulas are used to compute the substitutes property.

$$X^s_j(t+1)_2 = \left(X^s_j(t)\right)_2 - \theta\left(\left|\vartheta\left(X^s_j(t)\right)_2 - X^s_j(t)\right|\right) \tag{35}$$

$$X^s_j(t+1)_3 = \left(X^s_j(t)\right)_3 - \theta\left(\left|\vartheta\left(X^s_j(t)\right)_2 - X^s_j(t)\right|\right) \tag{36}$$

$$X^s_j(t+1) = \frac{X^s_j(t+1)_1 + X^s_j(t+1)_2 + X^s_j(t+1)_3}{3} \tag{37}$$

Up to this point, we have discussed learning process and expressed all notations related to this part of the algorithm. The following point needs to be considered:

- In order, a team gets closer to the best teams, the value of b is linearly decreased from β to 0. It must be noted that value for

Table 6
Pseudo-code for season transfers process.

For k=1 to N_{st}
  $H = \{ select\ randomly\ index\ i\ from\ N | i \notin H \}$

End For
For k=1 to N_{st}
  For j=1 to J
  r=rand()
    If r>0.5
      w=select randomly from current available teams
      $H_j^f(k) = w_j^f$
      $H_j^s(k) = w_j^s$
    End If
  End For
  $CostFunction\ (k) = f(X^f(k))$
End For

θ is in the range $-b\beta$ to βF and the next position of the current team can be anywhere within the search space. This is used in the algorithm to make a better exploitation.

2.10. Season transfers

In the volleyball league, the transfer is a process in which a player moves from one team to another. The players can be transferred at the end of the season with respect to some regulations. Based on this concept, we define an operator in the VPL to imitate the process and to help the algorithm to converge toward an optimal solution. In the season transfer time, we have a set $H \subset N$, consisting teams which have been chosen randomly. All positions of each member of set H is selected randomly from the current available teams, if $r > 0.5$, where r is a uniformly distributed random number [0,1]. We suppose that only a fraction of all teams uses the season transfers process. δ_{st} denotes the percentage of teams which participate in season transfers. Therefore, the number of teams participate in season transfer, N_{st}, is defined using Eq. (39):

$$N_{st} = \left[N\delta_{st} \right] \tag{38}$$

The pseudo-code to implement season transfers is shown in Table 6.

2.11. Promotion and relegation process

After a season is ended, top team (teams) is (are) moved up to the higher division league, and the worst team (teams) would go down to a lower division for the next season. This process is called promotion and relegation. The number of teams that move depends on the league regulations. Let δ_{pr} indicates the rate of promoted and relegated teams at the end of a season, and the following formula is defined to calculate the number of teams moved:

$$N_{pr} = \left[N\delta_{pr} \right] \tag{39}$$

Where N_{pr} denotes the number of teams moved to another league, and N represents the total number of teams in the league. In a common volleyball league, N_{pr} teams which have the lowest rank are selected to move to the lower division league. We simply supposed that only one league is available in the algorithm. We use a special process for determining the promoted team. In this way, N_{pr} teams with the worst values are removed from the league, and N_{pr} new teams are generated and added to the league. To generate each promoted team, the formation and substitutes properties are randomly selected from the formation and substitutes properties of teams in the current premier league, respectively. Based on the explanations, the pseudo-code for promotion and relegation process is presented in Table 7.

Table 7
Pseudo-code for the promotion and relegation process.

Remove N_{pr} worst teams of league.
Define N_{pr} empty teams NT with two properties: formation and substitutes
For k=1 to N_{pr}
  For j=1 to J
    s=select randomly from current available teams $(N - N_{pr})$
    $NT_j^f(k) = s_j^f$
    $NT_j^s(k) = s_j^s$
  End For
  $CostFunction\ (k) = f(NT^f(k))$
End For
Add NT teams to league

2.12. A comparison among VPL, SCL, and LCA

As earlier stated, it is important to determine the main differences between our new approach and the algorithms published in [108,109]. A deep comparison helps the readers to gain a better understanding of our proposed algorithm. Table 8 shows the main differences and similarities between the three algorithms with respect to various features, which clarifies our contributions to the proposed algorithm.

3. Experiments and results

In this section, the performance of VPL algorithm is investigated for 23 benchmark functions. Some of these test functions are used by many researchers such as [28,95,99–101,104,109]. We will use these test functions to compare the VPL algorithm with the other well-known metaheuristic algorithms. The test functions are organized into three categories: unimodal, multimodal and fixed-dimension multimodal benchmark functions. The unimodal functions (F1–F7), described in Table 9, are suitable for benchmarking the exploitation of algorithms since they have only one global optimum. These unimodal test functions are illustrated in 3-D diagrams in Fig. 7.

In the second category, multimodal functions (F8–F13), each have a massive number of local optima. They are extremely helpful to examine the exploration and local optima avoidance of algorithms. The details of this class of test functions are shown in Table 10. We can also see the 3-D diagrams of these functions as shown in Fig. 8.

Different features of the third category, fixed-dimension multimodal benchmark functions, including the cost function, range of variables and the optimal value, are summarized in Table 11 (Fig. 9).

In order to equalize the maximum number of function evaluations, we set only 100,000 evaluations for all algorithms, and also set 100 iterations for each run. As it is expressed [112,113], each function recurs 30 times, and results of VPL will be compared with some well-known algorithms. Before running these experiments, we must set some parameters, so the next section is devoted to the analysis of various parameters.

3.1. Effects of various parameter settings in VPL

Many parameters of evolutionary algorithms are defined as static, which means they are not altered while the optimization process is running. An important static parameter is the population size, which is a core concept in many evolutionary algorithms. The disadvantage of setting the parameters static is the lack of flexibility during the search process. So, one of the most challenging stages of implementing an evolutionary algorithm is parameters

Table 8
A comparison among SCL, LCA, and VPL with respect to various features.

features	SCL	LCA	VPL
Solution structure	includes just formation property	includes just formation property	includes both formations and substitution
population	League size	League size	League size
Iteration	week	week	season
Initialization	Randomly initializes the team formations according to the league size and the number of variables	Randomly initializes the team formations according to the league size and the number of variables	Randomly initializes the team formations and substitution according to the league size and the number of variables
League schedule	Using SRR	Using SRR	Using SRR
Match analysis	Based on team power	Using SWOT[a] analysis and yield four equation to find result of match.	Based and probability rules and team power index and a new linear equation
Learning phase	Not included	Not included	This operator moves teams toward the top 3 best teams
Season transfer	Not available	The number of dimension is randomly selected for this operator, and applied at the end of each season	Applying this operator based on the parameter δ_{st}, and it is applied at the end of each iteration (season)
Promotion and regulation	Promotion and regulation	Promotion and regulation	Promotion and regulation
internal competition in each team among players	Members of each team try to move toward key player	Not included	Not included
key player in each team	Included	Not included	Not included
super star player or best team in the league	super star player	Not included	Best team
Strategy for winner team	Imitation and provocation	Applying SWOT equations	Winner strategy
Strategy for the loser team	Not included	Applying SWOT equations	Knowledge sharing and repositioning strategy

[a] Strengths, Weaknesses, Opportunities and Threats (SWOT).

Table 9
Unimodal benchmark functions.

Function	Dim	Range	f_{min}
$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[−100, 100]	0
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[−10, 10]	0
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	30	[−100, 100]	0
$f_4(x) = max_i \left\{ \lvert x_i \rvert, 1 \le i \le n \right\}$	30	[−100, 100]	0
$f_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right]$	30	[−30, 30]	0
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[−100, 100]	0
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	[−1.28, 1.28]	0

tuning. In this section, we investigate the effect of various settings of control parameters on the performance of VPL. Since it is difficult to analyze and manipulate the effect of all input parameters on the performance of VPL concurrently, we will focus on the main parameters. We first examine the effect of different levels of δ_{pr}, and then the effects of N, δ_{st} and β will be investigated. In order to evaluate the performance of VPL extensively, an empirical study based on a set of 6 benchmark functions, selected from different categories (F1, F2, F8, F10, F12, F14, and F18) will be discussed.

The promotion and relegation rate are the first two parameters, which are analyzed in this section. To understand how different settings for δ_{pr} influence on the performance of VPL, different levels of these parameters are considered. Results are summarized in Table 12.

As it is shown in Table 12, different levels of δ_{pr} are ranked, such that any level with the best value gets the highest rank. The resulting rankings of different levels of δ_{pr} in different test functions are illustrated in Fig. 10.

Here, most settings make VPL capable to find the global optimum and there is no significant gap in the obtained results. It seems that $\delta_{pr} = 0.05$ is the most efficient setting since it has gained the best rank for F8 and F10 and the second rank for F2 and F18 twice. If we calculate the average ranks for all levels, results demonstrate that $\delta_{pr} = 0.5$ has the best performance in most cases. It could be seen that $\delta_{pr} = 0.05$ and $\delta_{pr} = 0.5$ are the proper settings which have better performance, compared with the other settings.

In order to analyze whether the number of teams has an important effect on the performance of VPL; different levels of the parameter are considered. The obtained results are summarized in Table 13.

As it can be clearly seen that $N = 10$ provides efficient results in most cases. According to Table 13, $N = 10$ has the best ranking in F1,

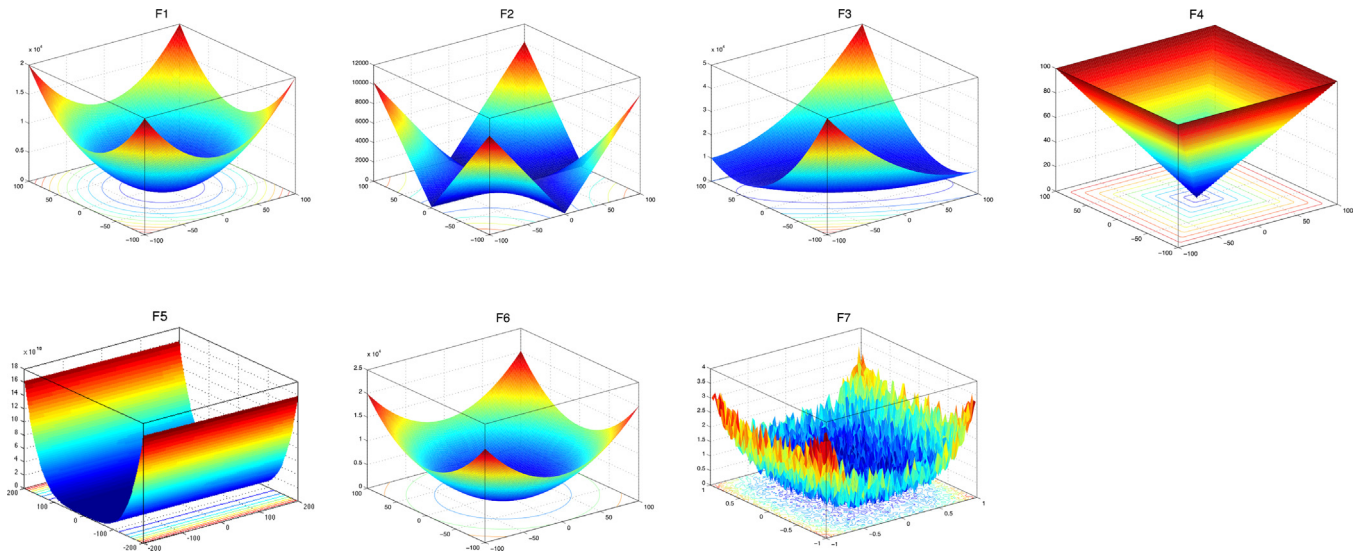

Fig. 7. 3-D diagrams of unimodal benchmark functions.

Table 10
Multimodal benchmark functions.

Function	Dim	Range	f_{min}
$f_8(x) = \sum_{i=1}^{n} -x_i \sin\sqrt{\lvert x_i \rvert}$	30	[−100, 100]	0
$f_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	[−10, 10]	0
$f_{10}(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	30	[−100, 100]	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[−100, 100]	0
$f_{12}(x) = \frac{\pi}{n}\left\{\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2\left[1+sin^2(\pi y_{i+1})\right] + (y_n-1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i-a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i-a)^m & x_i < -a \end{cases}$	30	[−30, 30]	0
$f_{13}(x) = 0.1\left\{\sin^2(3\pi y_1) + \sum_{i=1}^{n}(x_i-1)^2\left[1+sin^2(3\pi y_i+1)\right] + (x_n-1)^2\left[1+\sin^2(2\pi x_n+1)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[−100, 100]	0

F2, F12 and F18 test functions. Fig. 11 illustrates ranking of different levels in evaluating test functions.

As illustrated in Fig. 11, the VPL algorithm is capable to create much better ranking compared to the others. That is, $N = 10$ has the minimum value, so it will be selected as the best level of this parameter.

In order to investigate whether δ_{st} has a significant effect on the performance of VPL; different values for the parameter are considered as follows: 0, 0.18, 0.36, 0.54, 0.72, and 0.90. Results obtained under different levels are summarized in Table 14.

Resulting values for δ_{st} are calculated using different test functions. It can be seen that there is no dominant result in performing different levels of δ_{st}. Hence, if any level of the parameter has a better value during the optimization process, its rank will be the highest. Ranking of different levels of δ_{st} in different test functions is shown in Fig. 12.

Based on Fig. 12, it can be derived that the best performance is obtained when the value of this parameter is set to 0.54. Even though this parameter setting is not simply extracted from this figure, but as it justifies that the average rank indicates a superior performance than the others.

At the end of this section, we analyze the constant value β which indicates the learning phase of the proposed algorithm. We set this constant different levels 1–10. It can be simply observed from Table 15 that β is constant for functions F10 and F14. Ranking of different levels of this parameter for all test functions is depicted in Fig. 13.

The irregular behavior for different levels of β can be seen in Fig. 13. Apart from the test functions F10 and F14, all levels of the β result in obtaining optimum solutions. Results for test functions F8, F12 and F18 are surprisingly similar. There is no special pattern in test functions F1 and F2. However, if we calculate

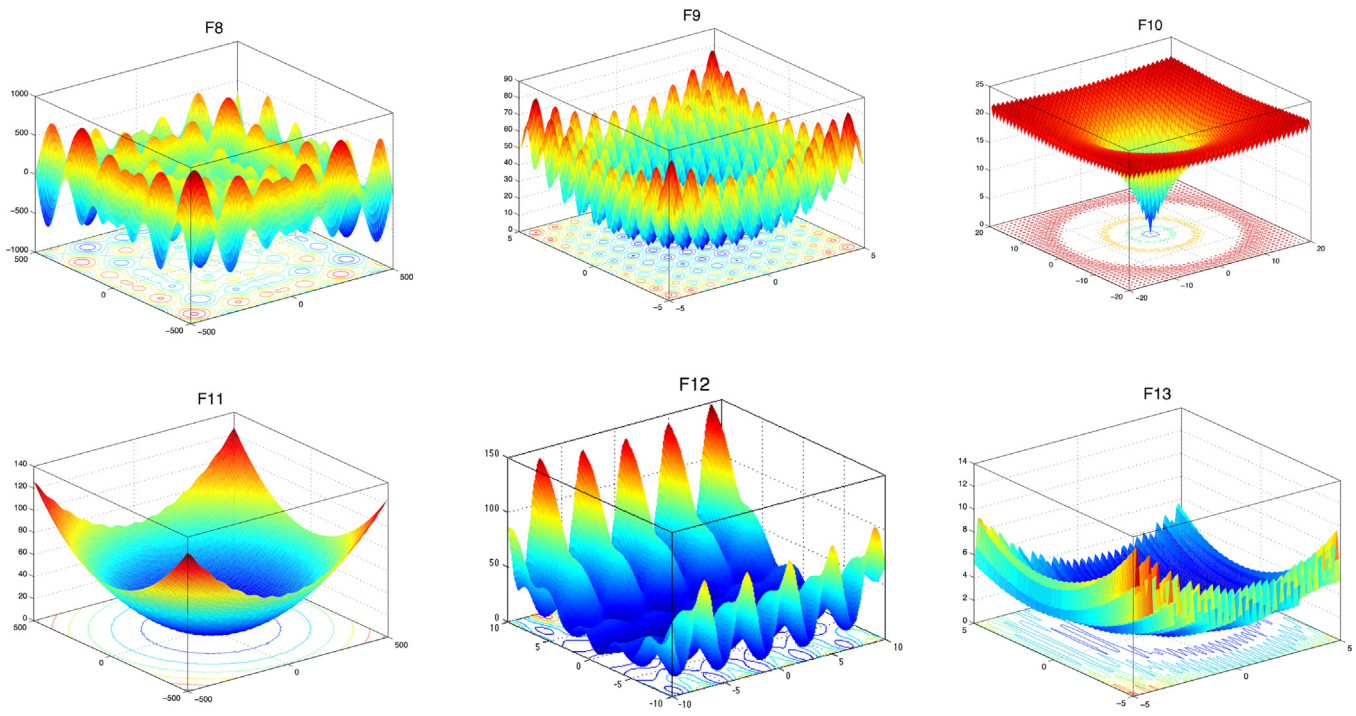

Fig. 8. 3-D diagrams of multimodal benchmark functions.

Table 11
Fixed-dimension multimodal benchmark functions.

Function	Dim	Range	f_{min}
$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} \left(x_i - a_{ij}\right)^6}\right)^{-1}$	2	$[-65, 65]$	1
$f_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1\left(b_i^2 + b_i x_2\right)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	$[-5, 5]$	0.00030
$f_{16}(x) = 4x_i^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]$	−1.0316
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cos(x_1) + 10$	2	$[-5, 5]$	0.398
$f_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right]. \left[30 + (2x_1 - 3x_2)^2.\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right]$	2	$[-2, 2]$	3
$f_{19}(x) = -\sum_{i=1}^{4} C_i exp\left(-\sum_{j=1}^{3} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	3	$[1, 3]$	−3.86
$f_{20}(x) = -\sum_{i=1}^{4} C_i exp\left(-\sum_{j=1}^{6} a_{ij}\left(x_j - p_{ij}\right)^2\right)$	6	$[0, 10]$	−3.32
$f_{21}(x) = -\sum_{i=1}^{5} \left[(X - a_i)(X - a_i)^T + C_i\right]^{-1}$	4	$[0, 10]$	−10.1532
$f_{22}(x) = -\sum_{i=1}^{7} \left[(X - a_i)(X - a_i)^T + C_i\right]^{-1}$	4	$[0, 10]$	−10.4028
$f_{23}(x) = -\sum_{i=1}^{10} \left[(X - a_i)(X - a_i)^T + C_i\right]^{-1}$	4	$[0, 10]$	−10.5363

the average ranking of different levels, $\beta = 7$ has the best performance.

Following the above discussion, we can determine the best parameter settings for the proposed algorithm. We use $L = 60$ and $pc = 0.5$ in our experiments. It would be generally concluded that proper parameter settings are $\delta_{pr} = 0.05$, $\delta_{st} = 0.54$, $N = 10$, and $\beta = 7$. We performed a simple method to tune our algorithm. Therefore, a comprehensive study may be required to find the best parameter setting, which is not the main concern of this paper.

3.2. Results from test functions

As it was mentioned before, we employ three sets of different problems explained above. In order to show the power of the

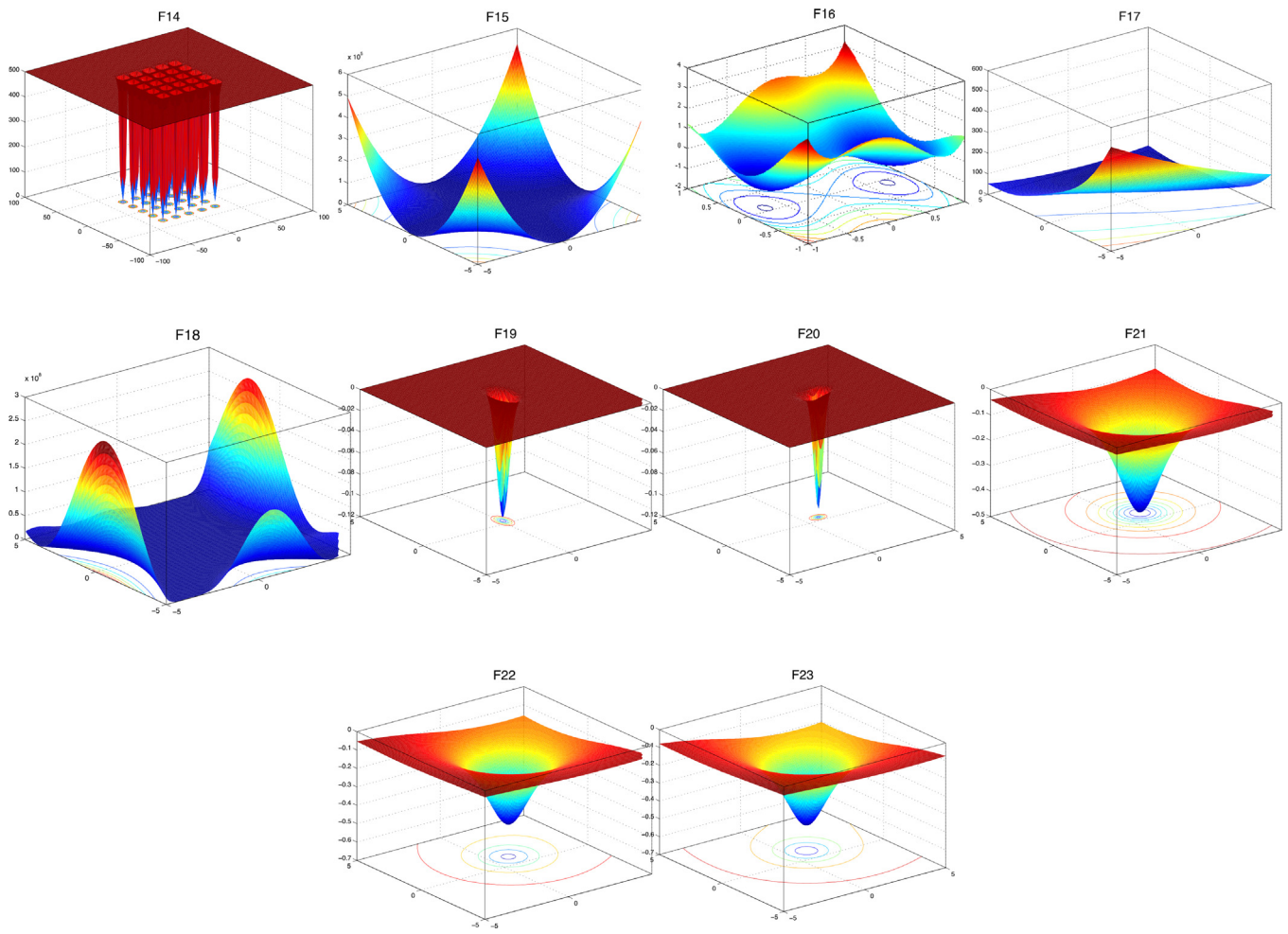

Fig. 9. 3-D diagrams of fixed-dimension multimodal benchmark functions.

Table 12
Results obtained from different levels of promotion and relegation rate (δ_{pr}).

δ_{pr}	F1	F2	F8	F10	F12	F14	F18
0	5.3E-150	2.8E-143	-5E+133	-2E+126	5.07E-85	1.8E-200	6.3E-233
0.05	3.7E-139	1.3E-156	-6E+133	-5E+128	2.1E-99	2.3E-189	5.8E-227
0.1	4.6E-143	2.3E-150	-3E+125	0.00	1.84E-85	1E-191	6.3E-205
0.15	2.8E-139	8.5E-150	-7E+124	1.3E-83	4.9E-106	5.2E-192	1.4E-216
0.2	9.2E-145	4.4E-142	-4E+125	1.26E-96	2.1E-124	3E-199	3.7E-217
0.25	9.5E-146	4.2E-138	-2E+128	5.83E-85	0.00	2.3E-213	3.4E-201
0.3	6.5E-144	4.7E-139	-9E+126	1.3E-104	8.2E-114	9.6E-223	3.5E-190
0.35	4E-148	2.4E-146	-4E+132	4E-124	6.4E-93	3.5E-213	4.6E-205
0.4	3.4E-145	3.4E-142	-3E+133	0.00	2.52E-76	2.1E-206	6.1E-200
0.45	9.1E-144	1.7E-159	-7E+126	4.1E-113	1.3E-133	1.2E-218	2.5E-214
0.5	5.8E-151	3.4E-149	-1E+130	1.44E-92	5.6E-137	3.2E-224	2.7E-219

Table 13
Results obtained from different levels of number of teams (N).

N	F1	F2	F8	F10	F12	F14	F18
10	2.748E-154	3.779E-150	-5.4E+128	1.175E-120	1.485E-130	1.335E-201	1.267E-238
20	3.37E-144	6.08E-134	-2.15E+128	3.72E-110	6.26E-01	3.00E-212	6.45E-207
30	1.07E-139	4.91E-140	-5.50E+131	2.43E-103	5.05E-01	6.35E-197	2.61E-211
40	6.90E-137	3.75E-135	-3.43E+130	1.65E-129	2.89E-01	9.07E-207	3.43E-213
50	9.62E-136	1.70E-154	-3.18E+125	7.90E-112	1.91E-119	1.51E-184	3.19E-197

proposed algorithm, we compared VPL with Particle Swarm Optimization (PSO), Differential Evolution (DE), Genetic Algorithm (GA), Artificial Bee Colony (ABC), Firefly Algorithm (FA), Harmony Search (HS), Sin Cosine Algorithm (SCA), Soccer League Competition (SLC), and League Championship Algorithm (LCA). In solving the aforementioned test functions, a total of 10 teams (and 10 search agents for the other algorithms) are used to determine the global optimum over 100 iterations. We analyze the unimodal functions (F1–F7) for

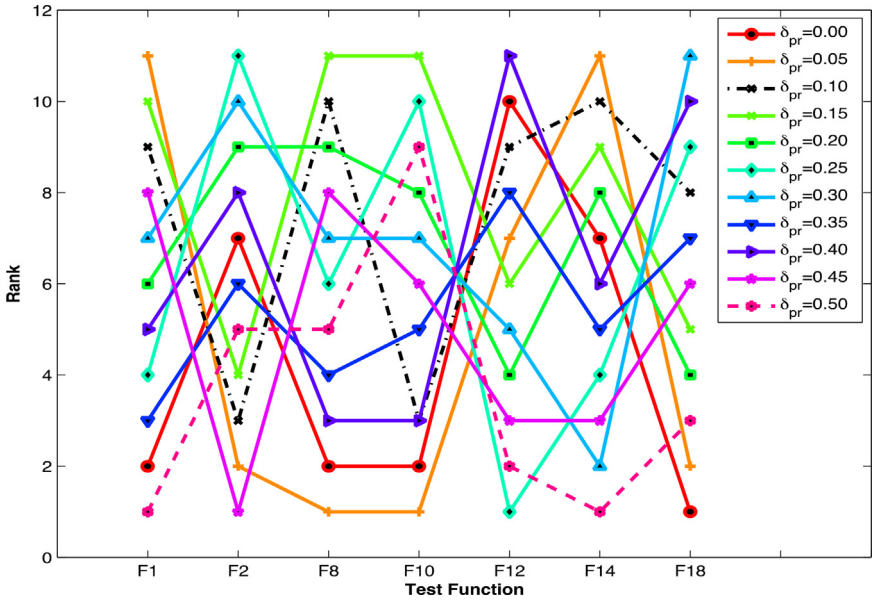

Fig. 10. Ranking of different levels of δ_{pr} for different test functions.

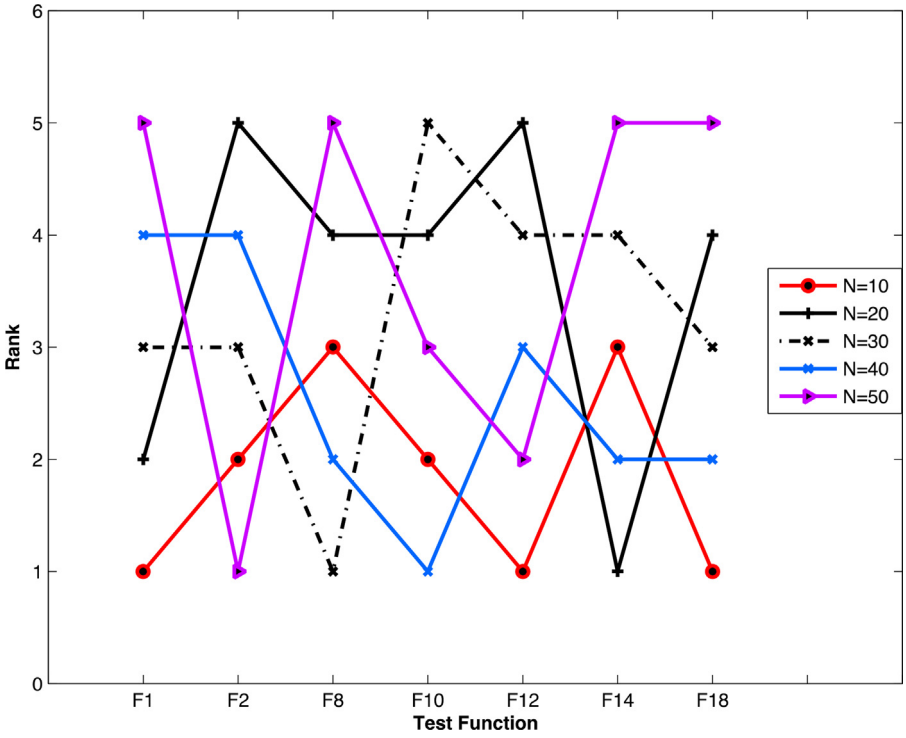

Fig. 11. Ranking of different levels of N in different test functions.

Table 14
Results obtained under different levels of δ_{st}.

δ_{st}	F1	F2	F8	F10	F12	F14	F18
0	1.09E-140	2.93E-147	2.32E-138	9.215E-137	9.96E-141	9.21E-136	1.09E-140
0.18	1.22E-145	1.95E-134	2.15E-144	1.725E-156	7.56E-146	1.72E-155	1.22E-145
0.36	2.91E-132	9.74E-147	9.22E-144	5.457E-151	3.65E-145	4.46E-150	2.91E-132
0.54	1.35E-145	1.88E-137	2.80E-136	1.39E-143	2.75E-153	1.39E-146	1.35E-145
0.72	9.22E-137	3.67E-139	1.12E-150	3.508E-140	2.20E-138	3.51E-141	9.22E-137
0.9	9.95E-146	4.32E-134	8.92E-135	2.488E-143	1.48E-147	2.49E-142	9.95E-146

measuring the exploitation capability of algorithms since these test functions have one global optimum and no local optima. Table 16 shows statistical results of unimodal benchmark functions.

According to Table 16, VPL has gained the best performance, and it has reached the first rank among all algorithms. The table shows that VPL has consistently performed better than its well-known

Table 15
Results obtained from different levels of β.

β	F1	F2	F8	F10	F12	F14	F18
1	1.819E-143	1.7244E-90	-6.61E+103	8.8818E-16	9.839E-06	0.99800384	3.00001642
2	2.323E-146	2.5816E-90	-5.946E+92	8.8818E-16	1.791E-05	0.99800384	3.00007477
3	1.798E-148	1.2683E-95	-1.1E+104	8.8818E-16	9.018E-06	0.99800384	3.00000131
4	4.239E-143	3.2137E-97	-9.9E+110	8.8818E-16	0.00015146	0.99800384	3.00003192
5	1.115E-134	3.432E-99	-1.751E+98	8.8818E-16	8.4412E-05	0.99800384	3.0000464
6	1.552E-133	7.6907E-93	-2.16E+102	8.8818E-16	1.1769E-05	0.99800384	3.00000486
7	5.628E-139	3.4461E-89	-3.684E+14	8.8818E-16	4.4787E-05	0.99800384	3.00000764
8	7.262E-138	6.5224E-95	-1.098E+98	8.8818E-16	5.3987E-06	0.99800384	3.00003802
9	3.031E-139	2.3435E-95	-1.62E+107	8.8818E-16	1.4433E-05	0.99800384	3.00002625
10	3.641E-149	3.1639E-95	-4.2E+108	8.8818E-16	4.0653E-05	0.99800384	3.00000001

Table 16
Results of unimodal benchmark functions.

Function		ABC	DE	FA	GA	HS
F1	Best	8.46E+03	8.08E+02	1.13E+01	1.47E+04	2.14E+04
	Worst	2.52E+04	2.83E+03	1.13E+01	3.87E+04	3.31E+04
	Mean	1.73E+04	1.65E+03	4.63E+00	2.60E+04	2.73E+04
	Std	4.79E+03	8.08E+02	2.15E+00	6.23E+03	2.89E+03
F2	Best	9.92E+01	9.75E+00	3.06E+00	5.11E+01	5.51E+01
	Worst	1.31E+07	1.98E+01	3.06E+00	2.24E+02	7.76E+01
	Mean	7.65E+05	1.40E+01	1.00E+00	8.80E+01	6.63E+01
	Std	2.53E+06	9.75E+00	5.58E-01	2.80E+01	5.84E+00
F3	Best	6.90E+04	4.21E+04	1.69E+04	3.47E+04	3.99E+04
	Worst	1.55E+05	9.57E+04	1.69E+04	8.83E+04	7.99E+04
	Mean	9.99E+04	6.61E+04	9.12E+03	5.64E+04	5.71E+04
	Std	2.43E+04	4.21E+04	2.90E+03	1.42E+04	9.22E+03
F4	Best	6.43E+01	5.00E+01	5.24E+01	6.99E+01	6.20E+01
	Worst	9.11E+01	7.48E+01	5.24E+01	8.83E+01	7.28E+01
	Mean	7.95E+01	6.17E+01	3.91E+01	8.11E+01	6.73E+01
	Std	6.35E+00	5.00E+01	6.93E+00	4.94E+00	2.41E+00
F5	Best	4.37E+07	8.83E+04	3.40E+04	1.71E+07	2.10E+07
	Worst	1.89E+08	5.15E+06	3.40E+04	1.23E+08	8.96E+07
	Mean	1.06E+08	9.80E+05	3.48E+03	5.60E+07	5.39E+07
	Std	3.51E+07	8.83E+04	6.16E+03	2.19E+07	1.15E+07
F6	Best	6.90E+03	7.40E+02	9.36E+00	1.23E+04	1.78E+04
	Worst	2.83E+04	6.99E+03	9.36E+00	3.44E+04	3.33E+04
	Mean	1.85E+04	2.05E+03	3.88E+00	2.52E+04	2.66E+04
	Std	4.62E+03	7.40E+02	1.41E+00	5.70E+03	3.78E+03
F7	Best	6.90E+03	5.23E-01	8.50E-01	6.23E+00	1.34E+01
	Worst	1.02E+02	3.06E+00	8.50E-01	7.11E+01	3.74E+01
	Mean	4.16E+01	1.02E+00	3.64E-01	2.58E+01	2.42E+01
	Std	1.78E+01	5.23E-01	1.43E-01	1.18E+01	5.26E+00

Function		PSO	SCA	SLC	LCA	VPL
F1	Best	1.12E+02	2.67E+02	1.90E-166	1.41E-48	0.00E+00
	Worst	7.23E+02	1.57E+04	1.26E-85	1.25E-45	2.34E-130
	Mean	2.89E+02	6.39E+03	4.40E-160	3.25E-46	7.81E-132
	Std	1.64E+02	3.98E+03	3.91E-80	1.79E-46	4.20E-131
F2	Best	3.56E+00	1.30E+00	1.12E-125	4.58E-25	1.12E-102
	Worst	1.53E+01	1.98E+01	1.05E-109	5.86E-24	2.85E-89
	Mean	6.85E+00	8.72E+00	8.85E-06	9.79E-25	1.13E-90
	Std	2.55E+00	4.45E+00	9.84E+03	1.49E-24	5.13E-90
F3	Best	2.34E+03	1.58E+04	2.58E-25	3.26E+03	1.93E-33
	Worst	1.11E+04	7.14E+04	8.16E-01	7.88E+03	1.53E-02
	Mean	5.74E+03	4.18E+04	2.11E-02	1.12E+03	8.16E-04
	Std	2.30E+03	1.33E+04	1.33E+01	6.06E+03	2.85E-03
F4	Best	1.32E+01	5.64E+01	8.96E-30	1.49E+00	0.00E+00
	Worst	3.11E+01	8.99E+01	9.97E-01	3.47E+00	1.63E-28
	Mean	2.02E+01	7.59E+01	3.07E-04	5.09E-01	1.54E-29
	Std	4.47E+00	7.78E+00	1.03E+00	2.63E+00	3.96E-29
F5	Best	1.89E+03	1.91E+06	3.96E+01	9.86E-03	2.58E+01
	Worst	5.26E+04	1.32E+08	3.00E+00	2.87E+00	2.67E+01
	Mean	1.59E+04	3.04E+07	3.00E+01	8.42E-01	2.62E+01
	Std	1.19E+04	3.28E+07	3.32E-01	7.15E-01	2.76E-01
F6	Best	5.53E+01	7.20E+02	1.02E+01	0.00E+00	1.82E-05
	Worst	7.50E+02	2.54E+04	1.04E+01	0.00E+00	2.34E-03
	Mean	2.51E+02	7.72E+03	1.05E+01	0.00E+00	4.09E-04
	Std	1.51E+02	5.51E+03	1.12E-01	0.00E+00	5.33E-04
F7	Best	1.05E-01	6.95E-01	3.43E-01	1.56E-02	4.67E-05
	Worst	6.62E-01	2.94E+01	3.34E-01	5.91E-02	4.81E-03
	Mean	3.14E-01	1.06E+01	3.76E-06	9.48E-03	1.93E-03
	Std	1.43E-01	7.19E+00	1.60E+01	3.44E-02	1.36E-03

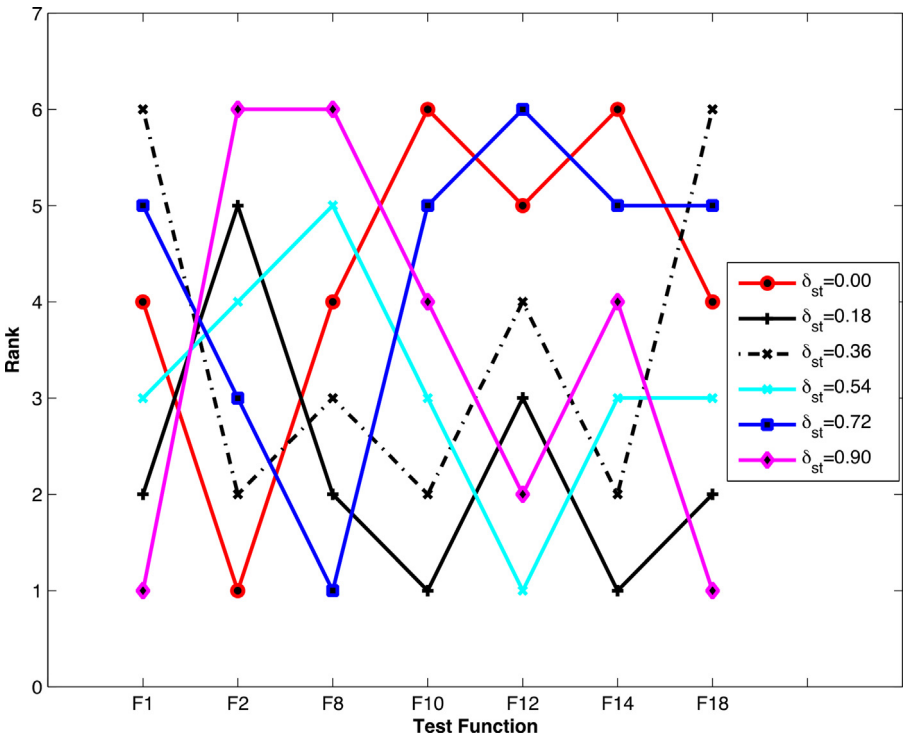

Fig. 12. Ranking of different levels δ_{st} in different test functions.

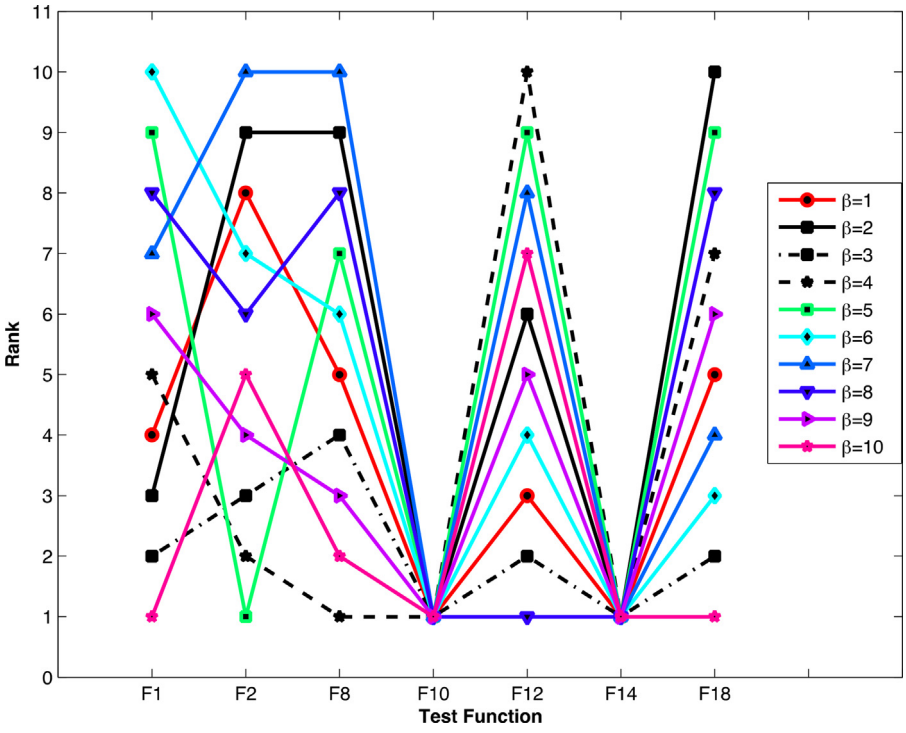

Fig. 13. Ranking of different levels of β in different test functions.

rival algorithms. In fact, VPL algorithm has excellent performance in exploitation and convergence, and it successfully overcomes to solve all the optimization problems within this category. After briefly reviewing results of the first category, we will focus on the second set of problems shown in Table 17.

Using test functions could be very useful while the exploration capability of an optimization algorithm is considered. From this viewpoint, results show that VPL is eligible for solving problems with challenging search space. In this case, VPL has shown very good performance in comparison with the other nine algorithms. In the last part of this analysis, fixed-dimension multimodal benchmark functions are taken into consideration. The statistical results of this category are shown in Table 18.

Inspecting the results, we can see promising results for this category. In the fixed-dimension multimodal functions category, they are designed to have many local optima where computa-

Table 17
Results of multimodal benchmark functions.

Function		ABC	DE	FA	GA	HS
F8	Best	-1.51E+21	-6.18E+04	-6.44E+03	-8.24E+03	-7.93E+03
	Worst	-5.74E+11	-1.59E+04	-6.44E+03	-5.40E+03	-5.39E+03
	Mean	-9.46E+19	-2.68E+04	-7.47E+03	-6.81E+03	-6.14E+03
	Std	2.96E+20	-6.18E+04	6.13E+02	6.80E+02	5.14E+02
F9	Best	2.79E+02	1.49E+02	2.00E+02	1.36E+02	2.41E+02
	Worst	4.01E+02	2.26E+02	2.00E+02	2.67E+02	2.89E+02
	Mean	3.45E+02	1.90E+02	1.16E+02	1.98E+02	2.64E+02
	Std	2.83E+01	1.49E+02	3.48E+01	3.17E+01	1.17E+01
F10	Best	1.66E+01	8.05E+00	4.66E+00	1.63E+01	1.80E+01
	Worst	2.05E+01	1.43E+01	4.66E+00	1.95E+01	1.94E+01
	Mean	1.90E+01	1.04E+01	2.55E+00	1.89E+01	1.88E+01
	Std	1.16E+00	8.05E+00	7.78E-01	6.12E-01	2.75E-01
F11	Best	7.54E+01	8.49E+00	1.09E+00	1.37E+02	1.81E+02
	Worst	2.92E+02	4.67E+01	1.09E+00	3.07E+02	3.13E+02
	Mean	1.48E+02	1.77E+01	1.03E+00	2.26E+02	2.51E+02
	Std	4.55E+01	8.49E+00	2.23E-02	3.65E+01	3.25E+01
F12	Best	7.23E+07	4.28E+01	6.33E+01	1.84E+07	3.25E+07
	Worst	7.40E+08	1.07E+07	6.33E+01	2.43E+08	1.17E+08
	Mean	2.31E+08	4.79E+05	2.31E+01	9.48E+07	7.99E+07
	Std	1.33E+08	4.28E+01	1.31E+01	6.08E+07	2.50E+07
F13	Best	2.48E+08	6.38E+04	4.05E+03	4.06E+07	1.04E+08
	Worst	8.64E+08	6.22E+06	4.05E+03	4.15E+08	2.84E+08
	Mean	4.94E+08	1.48E+06	2.03E+02	1.39E+08	2.03E+08
	Std	1.72E+08	6.38E+04	7.17E+02	7.88E+07	4.78E+07

Function		PSO	SCA	SLC	LCA	VPL
F8	Best	-7.79E+03	-3.51E+03	-1.33E-25	-3.72E+03	-1.19E+112
	Worst	-3.84E+03	-2.44E+03	4.47E-04	-1.06E+03	-7.38E+90
	Mean	-6.21E+03	-2.95E+03	-7.07E+03	7.53E+02	-4.68E+110
	Std	7.65E+02	3.19E+02	1.24E+02	2.21E+03	2.15E+111
F9	Best	3.24E+01	3.97E+01	0.00E+00	0.00E+00	0.00E+00
	Worst	1.51E+02	3.22E+02	0.00E+00	0.00E+00	0.00E+00
	Mean	8.96E+01	1.63E+02	0.00E+00	0.00E+00	0.00E+00
	Std	2.58E+01	6.38E+01	0.00E+00	0.00E+00	0.00E+00
F10	Best	3.54E+00	7.59E+00	7.08E-16	2.22E-14	8.88E-16
	Worst	7.88E+00	2.06E+01	5.18E-12	4.35E-14	8.88E-16
	Mean	5.57E+00	1.80E+01	7.05E-14	4.93E-15	8.88E-16
	Std	9.84E-01	3.96E+00	1.86E-29	3.74E-14	9.86E-32
F11	Best	1.55E+00	4.36E+00	0.00E+00	1.28E-13	0.00E+00
	Worst	8.55E+00	1.66E+02	0.00E+00	2.22E-02	0.00E+00
	Mean	3.60E+00	5.71E+01	0.00E+00	5.84E-03	0.00E+00
	Std	1.92E+00	4.19E+01	0.00E+00	2.65E-03	0.00E+00
F12	Best	5.36E+00	4.92E+04	1.04E+01	1.57E-32	1.11E-06
	Worst	3.72E+01	2.51E+08	1.05E+01	1.57E-32	6.56E-05
	Mean	1.41E+01	9.92E+07	0.00E+00	1.09E-47	2.58E-05
	Std	6.68E+00	7.19E+07	8.46E-82	1.57E-32	1.74E-05
F13	Best	2.45E+01	9.70E+05	7.61E-01	1.35E-32	2.63E-05
	Worst	3.48E+03	5.21E+08	9.20E-01	1.35E-32	2.31E-03
	Mean	4.87E+02	1.48E+08	1.00E+00	5.47E-48	4.18E-04
	Std	1.00E+03	1.24E+08	3.09E-01	1.35E-32	4.84E-04

tion complexity increases dramatically with the problem size. The results reported in Table 18 for functions F14–F23 indicate that VPL has also a very good exploration capability. In fact, VPL algorithm is always among the top three algorithms in all test problems. This ability comes from performing various mechanisms of exploration in VPL algorithm that leads the algorithm towards the global optimum. At the beginning of the algorithm, team cost changes abruptly, after that, it gradually converges. It can be seen that VPL is competitive in solving many optimization problems.

In order to perform a pairwise comparison, the Wilcoxon Signed-Rank Test [114] was used. To this aim, hypothesis test can be used to determine how the algorithm performs. The following null hypothesis (H_0) and the alternative hypothesis (H_1) are considered to clarify the performance of the two algorithms:

$$\begin{cases} H_0 : \mu_1 = \mu_2 \\ H_1 : \mu_1 \neq \mu_2 \end{cases}$$

The significant level of rejecting the null hypothesis is 0.01. Hence, the p-value would be considered as the main criteria to either reject or accept the null hypothesis (H_0). A large p-value shows weak evidence against the null hypothesis, while a small p-value indicates enough evidence to reject it. The p-values of each unrelated hypothesis have been calculated and shown in Table 19.

As shown in Table 19, most of the p-values are less than 0.01 indicating that null hypothesis (H_0) is strongly rejected at 99% significance level, and the alternative hypothesis (H_1) are accepted signifying that obtained optimal values of our proposed algorithm are different from those of the other algorithms. From the results of all algorithms in 30 independent runs, some obtained values make p-values too small. Therefore, in order to avoid showing very small positive numbers, we simply perform a fixed threshold value in which the p-values that are smaller than 1e-14 is replaced with zero. As shown in the aforementioned table, only four p-values are dropped below the considered fixed threshold, which is generally related to SCL and LCA algorithms.

Table 20 shows the comparison of rank sum values between VPL and the others. In the table, $R+$ is the sum of ranks for the problems in which the first algorithm (VPL) outperformed the second,

Table 18
Results of fixed-dimension multimodal.

Function		ABC	DE	FA	GA	HS
F14	Best	1.03E+00	9.98E-01	1.55E+01	9.98E-01	9.98E-01
	Worst	6.77E+00	1.17E+01	1.55E+01	2.99E+00	9.98E-01
	Mean	2.76E+00	3.49E+00	4.91E+00	1.28E+00	9.98E-01
	Std	1.33E+00	9.98E-01	4.36E+00	6.20E-01	1.25E-05
F15	Best	1.02E-03	1.16E-03	5.46E-03	1.80E-03	9.15E-04
	Worst	2.53E-03	1.59E-02	5.46E-03	1.02E+00	5.39E-03
	Mean	1.38E-03	4.22E-03	1.48E-03	7.97E-02	1.91E-03
	Std	3.62E-04	1.16E-03	1.03E-03	1.86E-01	1.04E-03
F16	Best	-1.03E+00	-1.03E+00	-2.15E-01	-1.03E+00	-1.03E+00
	Worst	-1.03E+00	-1.03E+00	-2.15E-01	3.56E+00	-1.03E+00
	Mean	-1.03E+00	-1.03E+00	-1.00E+00	-2.39E-01	-1.03E+00
	Std	7.68E-08	-1.03E+00	1.47E-01	9.85E-01	1.33E-03
F17	Best	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01
	Worst	4.34E-01	3.98E-01	3.98E-01	4.02E-01	3.99E-01
	Mean	4.07E-01	3.98E-01	3.98E-01	3.99E-01	3.98E-01
	Std	9.46E-03	3.98E-01	7.48E-07	1.13E-03	1.65E-04
F18	Best	3.02E+00	3.00E+00	3.00E+01	3.00E+00	3.00E+00
	Worst	6.68E+00	8.40E+01	3.00E+01	9.42E+01	3.02E+00
	Mean	3.82E+00	5.70E+00	3.90E+00	2.44E+01	3.00E+00
	Std	8.27E-01	3.00E+00	4.85E+00	2.89E+01	4.77E-03
F19	Best	-3.86E+00	-3.86E+00	-2.88E-01	-3.00E-01	-3.00E-01
	Worst	-3.60E+00	-9.19E-01	-2.88E-01	-3.00E-01	-2.99E-01
	Mean	-3.79E+00	-3.69E+00	-3.00E-01	-3.00E-01	-3.00E-01
	Std	6.26E-02	-3.86E+00	2.25E-03	0.00E+00	2.46E-04
F20	Best	-3.31E+00	-3.32E+00	-3.20E+00	-3.32E+00	-3.31E+00
	Worst	-3.08E+00	-3.20E+00	-3.20E+00	-3.19E+00	-3.15E+00
	Mean	-3.22E+00	-3.30E+00	-3.26E+00	-3.28E+00	-3.24E+00
	Std	5.99E-02	-3.32E+00	5.93E-02	5.28E-02	4.93E-02
F21	Best	-9.21E+00	-1.02E+01	-2.63E+00	-1.02E+01	-1.01E+01
	Worst	-1.70E+00	-2.54E+00	-2.63E+00	-2.60E+00	-2.42E+00
	Mean	-5.51E+00	-6.90E+00	-5.24E+00	-4.74E+00	-4.58E+00
	Std	2.16E+00	-1.02E+01	3.31E+00	2.98E+00	3.07E+00
F22	Best	-9.60E+00	-1.04E+01	-2.75E+00	-1.04E+01	-1.04E+01
	Worst	-2.84E+00	-2.22E+00	-2.75E+00	-2.66E+00	-2.71E+00
	Mean	-5.79E+00	-6.77E+00	-8.49E+00	-6.32E+00	-6.49E+00
	Std	1.62E+00	-1.04E+01	3.18E+00	3.31E+00	3.33E+00
F23	Best	-9.90E+00	-1.05E+01	-2.43E+00	-1.05E+01	-1.04E+01
	Worst	-2.36E+00	-2.42E+00	-2.43E+00	-1.67E+00	-2.29E+00
	Mean	-6.24E+00	-6.25E+00	-6.88E+00	-4.23E+00	-7.33E+00
	Std	2.09E+00	-1.05E+01	3.67E+00	3.11E+00	3.01E+00

Function		PSO	SCA	SLC	LCA	VPL
F14	Best	1.99E+00	9.98E-01	1.06E-04	9.98E-01	9.98E-01
	Worst	1.74E+01	1.36E+01	7.44E+02	9.98E-01	9.98E-01
	Mean	8.90E+00	5.70E+00	0.00E+00	3.33E-16	9.98E-01
	Std	4.40E+00	4.00E+00	9.86E-32	9.98E-01	2.32E-13
F15	Best	6.73E-04	9.80E-04	0.00E+00	9.95E-04	2.45E-05
	Worst	2.74E-02	6.85E-03	1.44E-24	3.35E-03	1.82E-03
	Mean	3.25E-03	2.61E-03	2.22E-03	4.84E-04	1.25E-03
	Std	5.97E-03	1.56E-03	6.66E-16	1.29E-03	3.08E-04
F16	Best	-1.03E+00	-1.03E+00	2.92E-04	-1.01E+00	-1.03E+00
	Worst	-1.03E+00	-1.00E+00	0.00E+00	-9.56E-04	-1.03E+00
	Mean	-1.03E+00	-1.03E+00	4.49E-04	3.26E-01	-1.03E+00
	Std	3.20E-09	8.03E-03	1.14E-06	4.50E-01	2.56E-06
F17	Best	3.98E-01	3.98E-01	0.00E+00	3.98E-01	3.98E-01
	Worst	3.98E-01	5.17E-01	5.82E-02	3.98E-01	3.98E-01
	Mean	3.98E-01	4.22E-01	1.78E-15	1.11E-16	3.98E-01
	Std	1.08E-10	2.76E-02	0.00E+00	3.98E-01	2.69E-06
F18	Best	3.00E+00	3.00E+00	1.93E-14	3.00E+00	3.00E+00
	Worst	3.00E+00	3.16E+00	4.02E-160	3.00E+00	3.00E+00
	Mean	3.00E+00	3.02E+00	4.81E-82	9.33E-07	3.00E+00
	Std	4.24E-09	3.47E-02	2.58E-152	3.00E+00	7.58E-05
F19	Best	-3.00E-01	-3.00E-01	-4.10E-77	-1.96E-01	-3.85E+00
	Worst	-3.00E-01	-3.00E-01	1.42E+01	-6.21E-12	-3.49E+00
	Mean	-3.00E-01	-3.00E-01	-1.36E-26	4.96E-02	-3.77E+00
	Std	0.00E+00	0.00E+00	1.27E-04	2.81E-02	9.37E-02
F20	Best	-3.32E+00	-3.01E+00	-8.60E+03	-3.00E+00	-3.32E+00
	Worst	-3.20E+00	-1.15E+00	0.00E+00	-5.18E-01	-3.20E+00
	Mean	-3.31E+00	-2.56E+00	8.88E-16	5.89E-01	-3.28E+00
	Std	4.04E-02	5.20E-01	0.00E+00	-1.54E+00	5.41E-02
F21	Best	-1.02E+01	-4.30E+00	3.00E-25	-3.40E+00	-1.02E+01
	Worst	-2.63E+00	-3.51E-01	5.67E-04	-2.23E-01	-5.06E+00
	Mean	-6.24E+00	-1.06E+00	9.97E-01	5.67E-01	-9.30E+00
	Std	3.51E+00	9.39E-01	8.89E-04	5.23E-01	1.90E+00

Table 18 (*Continued*)

Function		PSO	SCA	SLC	LCA	VPL
F22	Best	-1.04E+01	-5.34E+00	-1.03E+00	-2.09E+00	-1.04E+01
	Worst	-2.75E+00	-3.74E-01	3.98E-01	-2.91E-01	-5.09E+00
	Mean	-6.44E+00	-1.59E+00	3.00E+00	3.63E-01	-8.99E+00
	Std	3.53E+00	1.26E+00	3.00E-01	7.00E-01	2.35E+00
F23	Best	-1.05E+01	-4.26E+00	-3.27E+00	-2.06E+00	-1.05E+01
	Worst	-1.86E+00	-5.54E-01	-1.02E+01	-4.70E-01	-3.57E+00
	Mean	-6.45E+00	-1.74E+00	-1.04E+01	4.22E-01	-9.40E+00
	Std	3.86E+00	1.02E+00	1.05E+01	9.31E-01	2.28E+00

Table 19
p-Value obtained from Wilcoxon Signed-Rank Test.

	ABC	DE	FA	GA	HS	PSO	SCA	SCL	LCA
F1	2.02E-11	3.01E-11	1.02E-11	3.02E-11	1.02E-11	3.02E-11	3.02E-11	5.46E-09	3.02E-11
F2	1.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F3	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F4	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	8.48E-09	3.02E-11
F5	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F6	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	1.21E-12
F7	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	4.20E-10	3.02E-11
F8	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
F9	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	0	0
F10	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	0	6.81E-13
F11	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	1.21E-12	0	1.21E-12
F12	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.00E-11	1.21E-12
F13	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	8.48E-09	1.21E-12
F14	3.02E-11	0.007777	3.02E-11	3.02E-11	3.02E-11	3.01E-11	3.02E-11	1.21E-12	1.21E-12
F15	0.00557	5.09E-08	0.761828	4.08E-11	7.66E-05	0.706171	1.60E-07	8.97E-07	0.141278
F16	1.35E-10	0.000178	5.46E-09	3.02E-11	5.49E-11	2.43E-05	3.02E-11	1.21E-12	3.02E-11
F17	3.02E-11	4.11E-12	0.024157	4.50E-11	3.69E-11	3.02E-11	3.02E-11	2.37E-12	1.21E-12
F18	3.02E-11	5.10E-10	0.994102	9.92E-11	8.15E-11	3.69E-11	4.50E-11	2.88E-07	2.94E-09
F19	0.970516	8.20E-06	1.72E-12	1.21E-12	4.11E-12	1.21E-12	1.21E-12	1.21E-12	3.02E-11
F20	2.13E-05	0.012212	0.093341	0.000952	2.13E-05	6.53E-08	3.02E-11	0.00302	3.02E-11
F21	7.09E-08	0.000812	5.46E-09	3.50E-09	2.23E-09	0.09049	3.02E-11	1.21E-12	3.02E-11
F22	8.66E-05	3.16E-05	5.27E-05	6.05E-07	1.11E-06	0.455297	6.70E-11	1.21E-12	3.02E-11
F23	1.29E-06	3.01E-07	1.87E-07	2.23E-09	1.03E-06	0.501144	3.69E-11	1.72E-12	3.02E-11

Table 20
Statistical results obtained by all algorithms for 30 independent runs.

Test Functions		F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17	F18	F19	F20	F21	F22	F23
VPL vs ABC	R+	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1103**	491	**1365**	**1365**	**918**	**1203**	**1280**	**1181**	**1243**
	R-	465	465	465	465	465	465	465	465	465	465	465	465	465	465	727	**1339**	465	465	912	627	550	649	587
VPL vs DE	R+	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1095**	**1284**	**1169**	465	495	613	745	**1142**	**1197**	**1262**
	R-	465	465	465	465	465	465	465	465	465	465	465	465	465	735	546	661	**1365**	**1335**	**1217**	**1085**	688	633	568
VPL vs FA	R+	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**936**	**1310**	**1068**	**914**	**1365**	**801**	**1310**	**1189**	**1268**
	R-	465	465	465	465	465	465	465	465	465	465	465	465	465	465	894	520	762	916	465	1029	520	641	562
VPL vs GA	R+	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1362**	**1365**	**1361**	**1353**	**1365**	**1139**	**1315**	**1253**	**1320**
	R-	465	465	465	465	465	465	465	465	465	465	465	465	465	465	468	465	469	477	465	691	515	577	510
VPL vs HS	R+	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1183**	**1359**	**1363**	**1355**	**1365**	**1203**	**1320**	**1245**	**1246**
	R-	465	465	465	465	465	465	465	465	465	465	465	465	465	465	647	471	467	475	465	627	510	585	584
VPL vs PSO	R+	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**941**	629	465	467	**1365**	549	**1030**	**966**	**961**
	R-	465	465	465	465	465	465	465	465	465	465	465	465	465	465	889	**1201**	**1365**	**1363**	465	**1281**	800	864	869
VPL vs SCA	R+	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1365**	**1270**	**1365**	**1365**	**1361**	**1365**	**1365**	**1365**	**1357**	**1363**
	R-	465	465	465	465	465	465	465	465	465	465	465	465	465	465	560	465	465	469	465	465	465	473	467
VPL vs SCL	R+	**1310**	465	**1365**	**1305**	**1365**	**1365**	**1338**	465	915	915	915	**1365**	**1305**	**1365**	**1245**	**1365**	**1365**	**1262**	465	**1113**	**1365**	**1365**	**1365**
	R-	520	**1365**	465	525	465	465	492	**1365**	915	915	915	465	525	465	585	465	465	568	**1365**	717	465	465	465
VPL vs LCA	R+	**1365**	**1365**	**1365**	**1365**	465	465	**1365**	**1365**	915	**1365**	**1365**	465	465	465	**1015**	**1365**	465	514	**1365**	**1365**	**1365**	**1365**	**1365**
	R-	465	465	465	465	**1365**	**1365**	465	465	915	465	465	**1365**	**1365**	**1365**	815	465	**1365**	**1316**	465	465	465	465	465

while $R-$ is the sum of ranks for the opposite. Numbers in boldface indicate the algorithm that performs significantly better in each comparison.

As it can be seen from the results shown in Table 21, VPL obtains competitive results compared to other algorithms. For unimodal test functions, VPL beats all algorithms except the two algorithms SCL and LCA. Also, for the other two groups of test functions, VPL obtains very good results in Wilcoxon signed-rank test. Apart from LCA and SCL, VPL has significantly outperformed the other rivals with respect to the sum of the ranks in all test function groups. In summary, from the results of 23 test functions, it could be easily concluded that VPL is very efficient and produces very competitive results based on quality and reliability criteria. In fact, VPL strongly competes with the current state of the art algorithms.

In this section, we analyze the convergence of the algorithm. The convergences of VPL for different test functions are shown in Fig. 14. VPL algorithm shows two different convergence patterns. In the first pattern, the convergence curve of VPL decreases gradually over the solution process. This behavior is evident in F1 and F6. On the other hand, the algorithm tends to be accelerated while the number of iterations increases, then it sharply dropped. This special pattern is caused by the use of an adaptive mechanism in the learning phase of VPL algorithm. This mechanism leads the algorithm to explore unexplored regions of the search space at the beginning of

Table 21
Aggregate results of the Wilcoxon Signed-Rank Test.

Group	VPL vs #	ABC	DE	FA	GA	HS	PSO	SCA	SCL	LCA	Total	Percentage
unimodal	better	7	7	7	7	7	7	7	6	5	60	**95.24**
	as good as	0	0	0	0	0	0	0	0	0	0	0.00
	worse	0	0	0	0	0	0	0	1	2	3	4.76
multimodal	better	6	6	6	6	6	6	6	2	3	47	**87.04**
	as good as	0	0	0	0	0	0	0	3	1	4	7.41
	worse	0	0	0	0	0	0	0	1	2	3	5.56
fixed dimension multi modal	better	9	6	8	10	10	6	10	9	7	75	**83.33**
	as good as	0	0	0	0	0	0	0	0	0	0	0.00
	worse	1	4	2	0	0	4	0	1	3	15	16.67

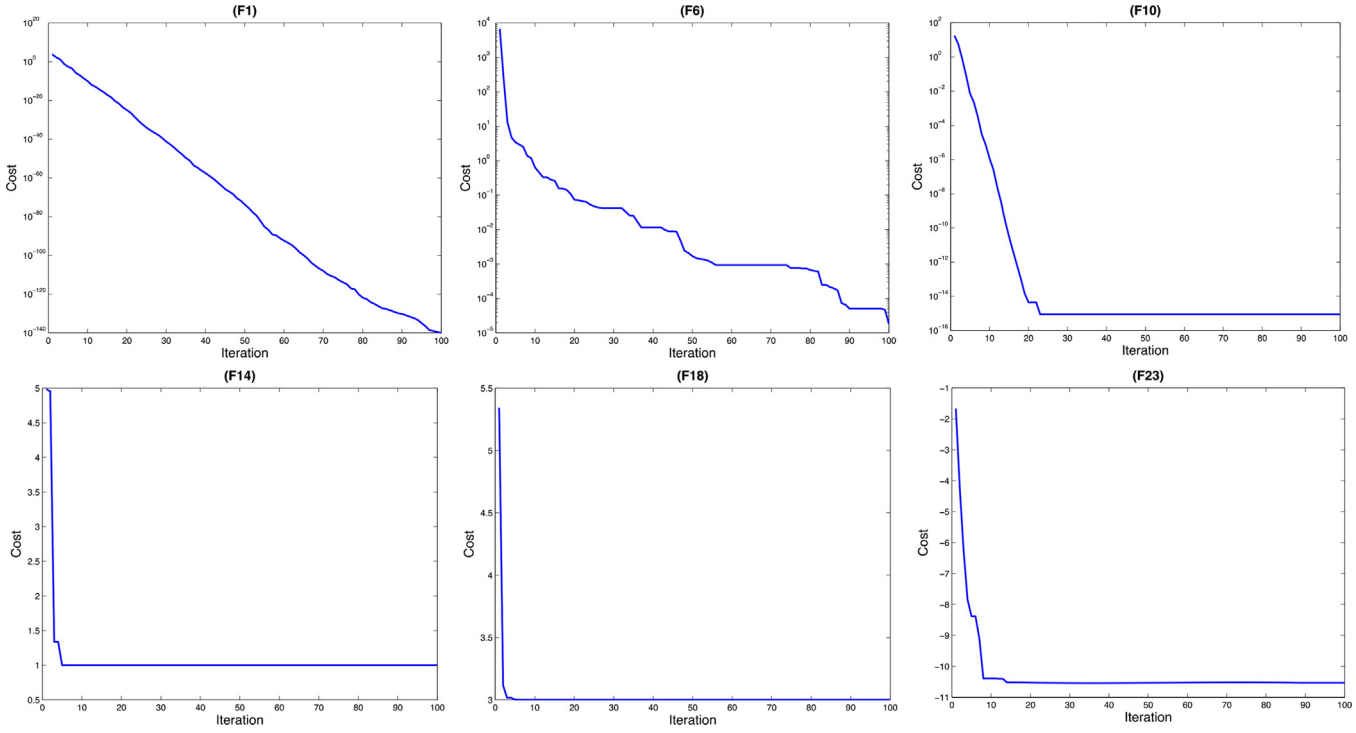

Fig. 14. The convergence curve of VPL.

the solution process and converges toward the optimum point very quickly. This behavior can be seen in F10, F14, F18, and F23.

In order to show how all search agents (teams) improve their results, the average fitness of all search agents during the optimization process is shown in Fig. 15. As it is seen, the behavior of average fitness of all agents is remarkably similar to convergence curve (see Fig. 14). The descending trend is quite evident in the convergence curve of VPL. It shows the capability of VPL algorithm in achieving very good results during the solution process.

The behavior of the search agents in solving test problems is shown in Fig. 16. There are abrupt fluctuations in the early steps of the process, and then it converges toward the best solution. This behavior clearly shows that each team first explores different points in the search space and then moves towards the best solution. It justifies the exploration power of the algorithm. It worth noting that in Fig. 15 all search agents (teams) are remarkably improved despite many fluctuations illustrated in Fig. 16.

Even though this section shows that VPL algorithm is promising to find the global optima of test functions, but there are some differences between benchmark functions and real-world optimization problems. The search space of these two classes of problems is completely different. There is a large number of constraints, both equality, and inequality, in real world problems. Therefore, it is worthwhile to examine the performance of VPL algorithm for solving constrained optimization problems in which the global optimum and search space are complex. This is the main concern of the next section, in which the three well-known engineering problems are solved using VPL algorithm.

4. Applications of VPL in classical engineering problems

In this section, we will solve three engineering design problems using VPL, and the results will be compared with the results already published in the literature. In order to handle the constraints, the penalty approach is used. The static penalty function is used in which penalty factors is not changed during the solution process. The following formula is used to calculate new objective function [115]:

$$F\left(\vec{x}\right) = f\left(\vec{x}\right) + \sum_{i=1}^{m}\left(\alpha_{k,i} \times max\left[0, g_i\left(\vec{x}\right)\right]^2\right) \tag{40}$$

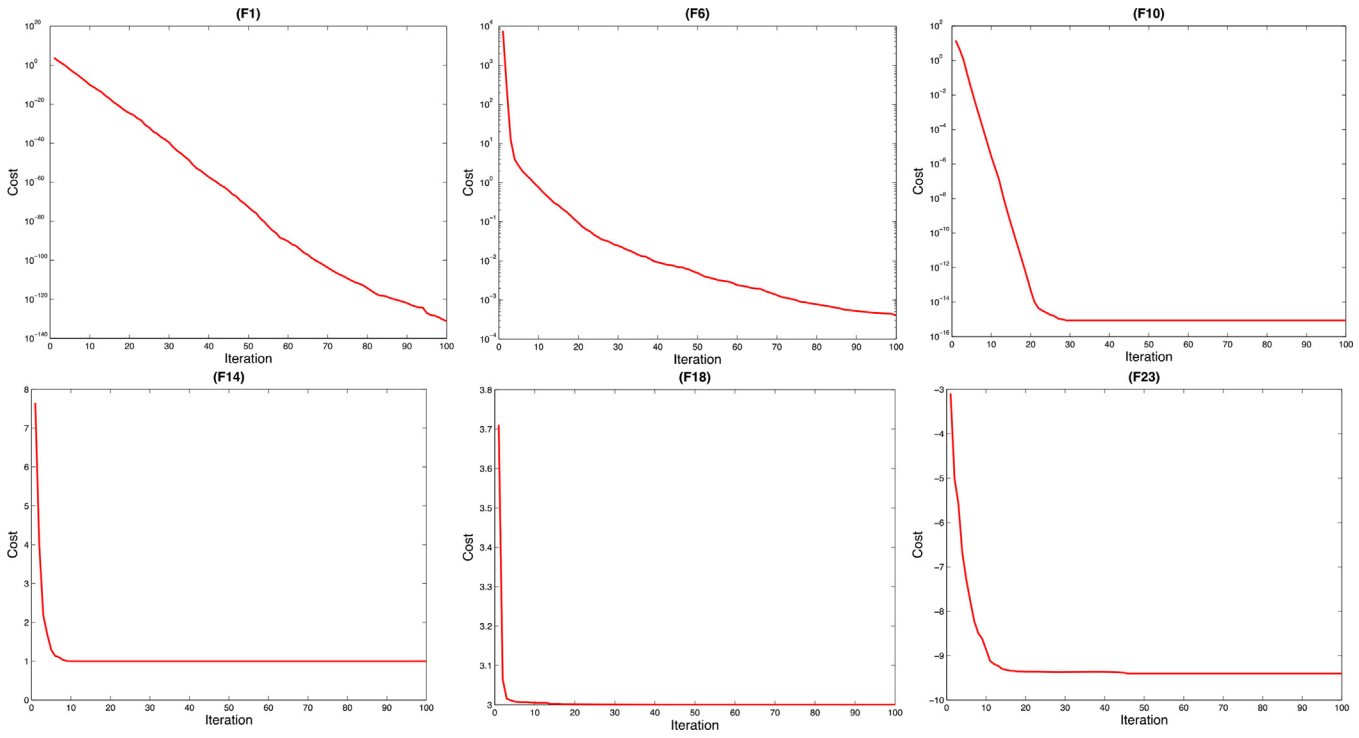

Fig. 15. Average fitness of teams.

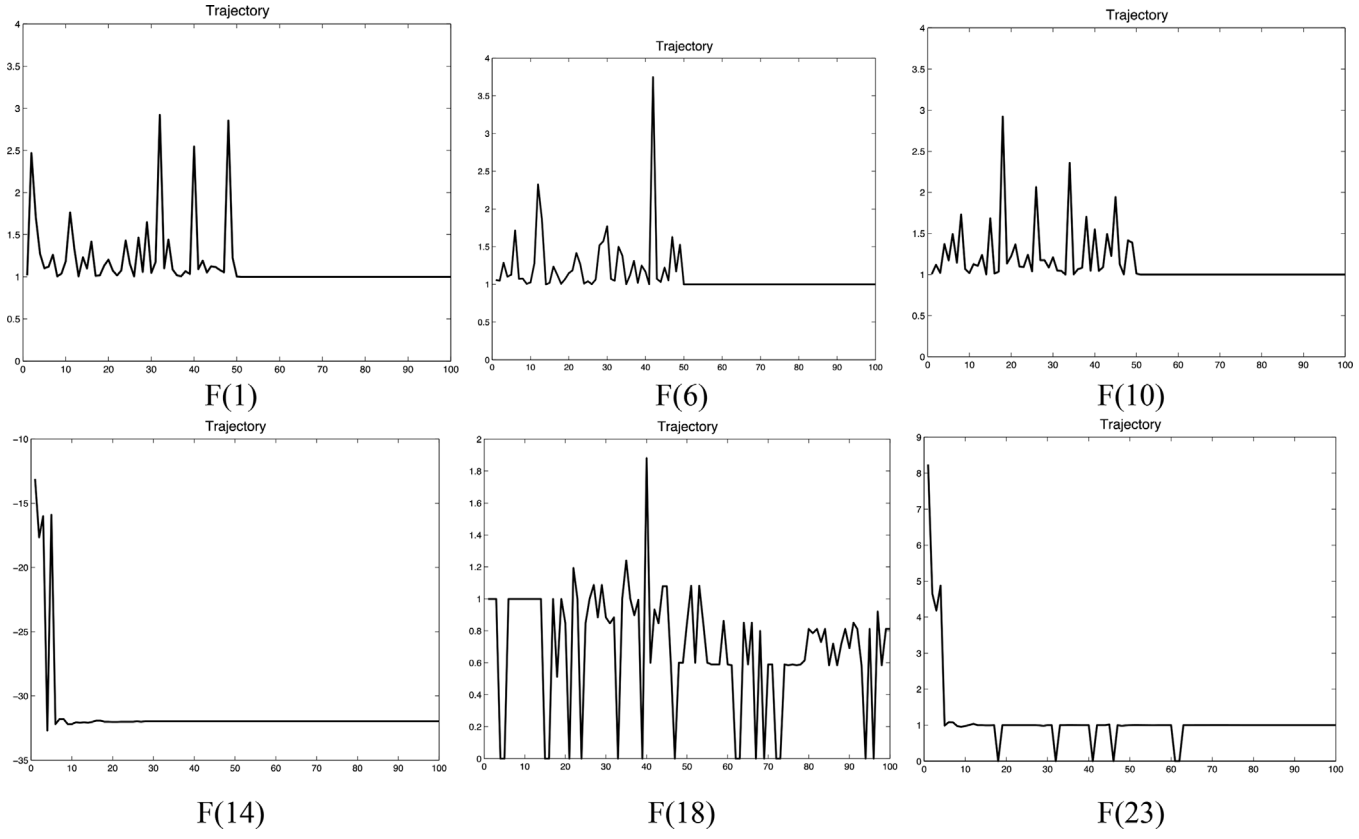

Fig. 16. Trajectory of the first variable of the first search agent when solving the test problems.

Where $\alpha_{k,i}$ denotes the penalty coefficient, $g_i\left(\vec{x}\right)$ denotes the violation from constraint i, m is the total number of constraints, and $f\left(\vec{x}\right)$ is the original objective function. All the optimum parameters calculated in the previous section are used to solve the problem.

4.1. Tension/compression spring design

In this problem, the objective is to minimize the weight of a tension/compression spring. This problem has been extensively

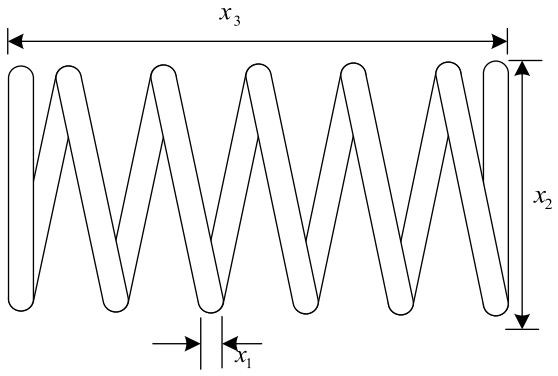

Fig. 17. Tension/compression spring design and its features.

studied in the literature. Constraints such as shear stress, surge frequency, and minimum deflection are considered. The schematic illustration of the problem is shown in Fig. 17.

The problem consists of three variables: wire diameter (d), mean coil diameter (D), and the number of active coils (N). These variables are shown in Fig. 17 as x_1, x_2 andx_3, respectively. The mathematical formulation of the problem is written as follows:

$$Consider \vec{x} = [x_1 x_2 x_3] = [dDN] \tag{41}$$

$$Minimizef\ (\vec{x}) = (x_3 + 2) x_2 x_1^2 \tag{42}$$

$$g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \tag{43}$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1 x_2}{12566\left(x_2 x_1^3 - x_1^4\right)} + \frac{1}{5108 x_1^2} - 1 \leq 0 \tag{44}$$

$$g_3(\vec{x}) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0 \tag{45}$$

$$g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \tag{46}$$

$$0.05 \leq x_1 \leq 2.00 \tag{47}$$

$$0.25 \leq x_2 \leq 1.30 \tag{48}$$

$$2.00 \leq x_3 \leq 15.00 \tag{49}$$

In the above formulation, the objective function expresses the minimization of the weight of a tension/compression spring, which is defined in Eq. (42). All constraints of the problem are considered in Eqs. (43)–(46), and finally Eqs. (47)–(49) define variable ranges.

This problem has been solved using various methods by different researchers. Coello [116], Belegundu [117], Coello and Montes [118], He and Wang [119], Montes and Coello [120], Kaveh and Talathari [121], Mahdavi [122], Li [123], and Kaveh [60] proposed different approaches to solve the problem. The comparison results of implementing different approaches are shown in Table 22. The results show that VPL has been able to find very good results compared to the others.

4.2. Pressure vessel design

In this problem, there is a special cylindrical vessel, which is capped at both ends by hemispherical heads. Similar to many other optimization problems, this problem is aimed to minimize the total cost, including the cost of the materials, forming and welding. The schematic illustration of the problem and its features are shown in Fig. 18.

There are four design variables: the thickness of the shell (x_1), the thickness of the head (x_2), the inner radius (x_3), and the length of the cylindrical section of the vessel (x_4). These variables are noted

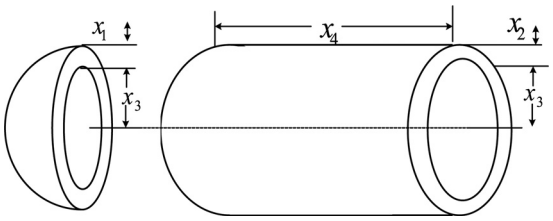

Fig. 18. Pressure vessel design and its features.

in the literature as T_s, T_h, R and L, respectively. This optimization problem can be written as follows:

$$Consider \vec{x} = [x_1 x_2 x_3 x_4] = [T_s T_h R L] \tag{50}$$

$$Minimizef\ (\vec{x}) = 0.62224 x_1 x_3 x_4 + 1.7781 x_2 x_3^2 + 3.1661 x_4 x_1^2 + \\ +19.84 x_3 x_1^2 \tag{51}$$

$$g_1(\vec{x}) = -x_1 + 0.0193 x_3 \leq 0 \tag{52}$$

$$g_2(\vec{x}) = -x_3 + 0.00954 x_3 \leq 0 \tag{53}$$

$$g_3(\vec{x}) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0 \tag{54}$$

$$g_4(\vec{x}) = x_4 - 240 \leq 0 \tag{55}$$

$$0 \leq x_1 \leq 99, \tag{56}$$

$$0 \leq x_2 \leq 99, \tag{57}$$

$$10 \leq x_3 \leq 200, \tag{58}$$

$$10 \leq x_4 \leq 200, \tag{59}$$

In the above formulation, the objective function expresses the minimization of the total cost as it is defined in Eq. (51). All constraints of the problem are considered in Eqs. (52)–(55), and finally Eqs. (56)–(59) are related to variables range.

In the literature, various kinds of methods have been applied to solve this problem. Those approaches can be mentioned as branch and bound method [124], augmented Lagrangian multiplier approach [125], different genetic algorithms [126], [116] [118], CPSO [119], ES [120], improved ACO [121], CSS [127], improved HS [122], and DE [123]. The results of the best solution found by different methods are presented in Table 23. Compared to the other techniques, VPL algorithm has found the better solution.

4.3. Welded beam design

The last example deals with the design of welded beam in which the overall fabrication cost should be minimized subject to constraints on shear stress (τ), buckling load on the bar (P_c), end deflection of the beam ($\triangleq$), and bending stress in the beam (σ). The schematic of the problem is illustrated in Fig. 19.

The problem has four design variables: the width $h(x_1)$ and length $l(x_2)$ of the welded area, the depth $t(x_3)$, and the thickness $b(x_4)$ of the main beam. The problem can be written as follows:

$$Consider \vec{x} = [x_1 x_2 x_3 x_4] = [hltb] \tag{60}$$

$$Minimizef\ (\vec{x}) = 1.10471 x_2 x_1^2 + 0.04811 x_3 x_4 (14.0 + x_2) \tag{61}$$

$$g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \leq 0 \tag{62}$$

$$g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \leq 0 \tag{63}$$

$$g_3(\vec{x}) = \delta(\vec{x}) - \delta_{max} \leq 0 \tag{64}$$

Table 22
Comparison of VPL results with literature for Tension/compression spring design.

Algorithm	Optimum variables			Optimum cost	Feasibility (Yes/No)
	d	D	N		
VPL	0.0501910	0.331680	12.834269	0.0123947	Yes
WOA	0.051207	0.345215	12.004032	0.0126763	Yes
PSO	0.051728	0.357644	11.244543	0.0126747	Yes
ES	0.051989	0.363965	10.890522	0.0126810	Yes
GA (Coello)	0.051480	0.351661	11.632201	0.0127048	Yes
IHS	0.051154	0.349871	12.076432	0.0126706	Yes
RO	0.051370	0.349096	11.76279	0.0126788	No
DE	0.051609	0.354714	11.410831	0.0126702	Yes
GA(Montes and Coello)	0.051643	0.355360	11.397926	0.0126698	Yes
ACO	0.051865	0.361500	11.00000	0.0126432	Yes
MCSS (Share et al. 2013)	0.051645	0.356496	11.271529	0.0126192	No

Table 23
Comparison of VPL results with literature for Pressure vessel design problem.

Algorithm	Optimum variables				Optimum cost	Feasibility (Yes/No)
	T_s	T_h	R	L		
VPL	0.815200	0.426500	42.0912541	176.742314	6044.9565	Yes
WOA	0.812500	0.437500	42.0982699	176.638998	6059.7410	Yes
improved HS	1.125000	0.625000	58.29015	43.69268	7197.730	No
CPSO	0.812500	0.437500	42.091266	176.746500	6061.0777	Yes
GA(Coello)	0.812500	0.437500	40.323900	200.000000	6288.7445	Yes
GA (Coello and Montes)	0.937500	0.437500	42.097398	176.654050	6059.9463	Yes
GA (Deb and Gene)	0.812500	0.437500	48.329000	112.679000	6410.3811	No
ES	0.812500	0.437500	42.098087	176.640518	6059.7456	No
DE	0.812500	0.437500	42.098411	176.637690	6059.7340	Yes
CSS	0.812500	0.437500	42.103624	176.572656	6059.0888	No
Branch and Bound	1.125000	0.625000	47.7000	117.7010	8129.1036	No
ALM	1.125000	0.625000	58.2910	43.69	7198.0428	No
GAS	0.937500	0.50000	48.329	112.679	6410.3811	Yes
ACO	0.812500	0.437500	42.098353	176.637751	6059.7258	Yes

Table 24
Comparison of VPL results with the other algorithms for the welded beam design problem.

Algorithm	Optimum variables				Optimum cost	Feasibility (Yes/No)
	h	l	t	b		
VPL	0.215235	6.898945	8.815033	0.216253	2.264711	Yes
WOA	0.205396	3.484293	9.037426	0.206276	1.730499	No
GSA	0.182129	3.856989	10.000000	0.202376	1.879952	No
CBO	0.205722	3.47041	9.037276	0.205735	1.724663	No
MCSS	0.205729	3.470493	9.036623	0.205729	1.724853	No
ACO	0.205700	3.471131	9.036683	0.205731	1.724918	No
CSS	0.205820	3.468109	9.038024	0.205723	1.724866	No
RO	0.203687	3.528467	9.004233	0.207241	1.735344	No
Improved HS	0.20573	3.47049	9.03662	0.2057	1.7248	No
GA (Deb)	0.2489	6.1730	8.1789	0.2533	2.4331	Yes
HS	0.2442	6.2231	8.2915	0.2443	2.3807	No
Random	0.4575	4.7313	5.0853	0.6600	4.1185	Yes
Simplex	0.2792	5.6256	7.7512	0.2796	2.5307	No
David	0.2434	6.2552	8.2915	0.2444	2.3841	Yes
APPROX	0.2444	6.2189	8. 2915	0.2444	2.3815	Yes
GA (Coello)	0.2489	3.420500	8.997500	0.21000	1.748309	No
GA (Coello and Montes)	0.205986	3.471328	9.020224	0.206480	1.728226	No
GA(He and Wang)	0.202369	3.544214	9.048210	0.205723	1.728024	No
ES	0.199742	3.612060	9.037500	0.206082	1.737300	No

$$g_4\left(\vec{x}\right) = x_1 - x_4 \le 0 \quad (65)$$

$$g_5\left(\vec{x}\right) = P - P_c\left(\vec{x}\right) \le 0 \quad (66)$$

$$g_6\left(\vec{x}\right) = 0.125 - x_1 \le 0 \quad (67)$$

$$g_7\left(\vec{x}\right) = 1.10471x_1^2 + 0.04811x_3x_4\left(14.0 + x_2\right) - 5.0 \le 0 \quad (68)$$

$$0.10 \le x_1 \le 2.00, \quad (69)$$

$$0.10 \le x_2 \le 10.00, \quad (70)$$

$$0.10 \le x_3 \le 10.00, \quad (71)$$

$$0.10 \le x_4 \le 2.00, \quad (72)$$

where

$$\tau\left(\vec{x}\right) = \sqrt{\left(\tau'\right)^2 + 2\tau'\tau''\frac{x_2}{2R} + \left(\tau''\right)^2}$$

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

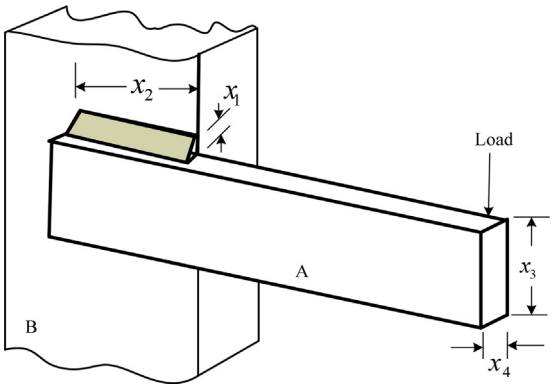

Fig. 19. Welded beam design and its features.

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$

$$\sigma\left(\vec{x}\right) = \frac{6PL}{x_4x_3^2}, \delta\left(\vec{x}\right) = \frac{6PL^3}{Ex_4x_3^2}$$

$P = 6000\text{lb}, L14\text{In}, \delta_{max} = 0.25\text{In}$

$E = 30 \times 10^6\text{psi}, G = 30 \times 10^6\text{psi}$

$\tau_{max} = 13600\text{psi}, \sigma_{max} = 30000\text{psi}$

In the above formulation, the objective function is defined in Eq. (61), all constraints of the problem are considered in Eqs. (62)–(68), and variables are defined Eqs. (69)–(72).

Regsdell and Philip [128] perform classical approaches like simplex, Davidon–Fletcher–Powell method, Deb [129], and Coello [116] to solve the problem. Coello and Montes [118] used GA in their work. Other methods used to solve this problem are ES [120], ACO [121], CSS [127], HS [24], improved HS [122], WOA [29] and MCSS [130] as shown in Table 24. There are some methods (e.g. WOA, GSA, CBO, MCSS, ACO) which have reported better objective function values than VPL algorithm. However, by checking the results, we have found that most of those results violate constraints of the model, resulting in infeasible solutions. The right-hand side column of Table 24 indicates whether the result is feasible. It is clear that VPL algorithm has been able to find the feasible optimal design for the problem.

5. Conclusion

As an alternative method for solving optimization problems, a novel population-based optimization algorithm is proposed in this research. In the proposed VPL algorithm, each solution is presented as a team in volleyball premier league, where each team tries to achieve the best position in the league table. To achieve that, each team requires updating its position using some special procedures related to the volleyball game. We formulated these behaviors as mathematical expressions. The formation of each team is updated based on its power index. The coach of the team arranges team members according to the situation of the team and the rival arrangement. This is the main point of this algorithm in which exploitation and exploration mechanisms are taken into account using the coaching concept. Another step of this algorithm mimics the competition process where two teams compete to achieve a better position in the league table. The coach also investigates the rival team's strengths and weaknesses, and usually, focuses on the best teams in the league to try to learn from their capabilities. We use this concept as the learning phase of the algorithm, in which teams update their properties according to top teams.

Additionally, two new mechanisms, namely season transfer, and promotion and relegation process, have been embedded in the algorithm to enhance similarity of VPL algorithm with its original concept. These two mechanisms improve the algorithm performance. Moreover, the effect of the control parameters on the performance of VPL was examined via implementing a comprehensive computational experiment, and then several experiments were done in order to verify the performance of VPL algorithm. Firstly, the set of well-known test problems including unimodal, multimodal, and fixed-dimension multimodal were employed to test exploration, exploitation, local optima avoidance, and convergence of the proposed algorithm. Secondly, three well-known engineering design problems were chosen and solved, and then the results were compared with results published in the literature. In all the experiments, results show that the performance of the proposed algorithm is significantly better than its rivals.

As we mentioned before, our proposed approach has been inspired by the general ideas and rules defined by two recently algorithms named LCA and SCL. In those algorithms, some common operators used to achieve better exploitation and exploration. To determine how these methods compete against each other, we investigate the main differences between operators.

All the other algorithms (except VPL) use similar solution structure, which is very different from the structure used in VPL. In addition, the other two algorithms use league size as one of the main parameters to determine typical population size as used in many population-based meta-heuristic algorithms. Similarly, they perform the same procedure, single round robin (SRR), to obtain league schedule. With regard to enhancing the exploitation and exploration in search space, promotion and regulation are also performed in all considered algorithms with the general framework known in sports. In VPL, we designed a winner strategy process to determine winner team while imitation an SWOT analyses applied in SCL and LCA, respectively. The authors of this article strongly believe that main advantage of the proposed algorithm comes from the learning phase, making all teams follow the top three teams. This operator mimics the analysis that the coach uses to determine what the leading teams in the league do to have an ideal performance. To our best of knowledge, the concept of this operator has not been applied in any other metaheuristic algorithm, yet. The learning phase creates an extensive searching range for the proposed algorithm in comparison with its rivals. The source code of VPL is publicly available at *bitbucket.org/salimifard/vpl*.

The authors would like to extend VPL theory and applications in the future researches. Firstly, different methods of parameter tuning can be investigated, and the binary version of VPL algorithm is worth researching. In addition, Multi-Objective Volleyball Premier League (MOVPL) algorithm and Discrete Volleyball Premier League (DVPL) algorithm are in developing phase.

References

[1] A. Sadollah, et al., Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems, Appl. Soft Comput. 13 (5) (2013) 2592–2612.
[2] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey Wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61.
[3] M. Sayadi, R. Ramezanian, N. Ghaffari-Nasab, A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems, Int. J. Ind. Eng. Comput. 1 (1) (2010) 1–10.

[4] T. Loukil, J. Teghem, D. Tuyttens, Solving multi-objective production scheduling problems using metaheuristics, Eur. J. Oper. Res. 161 (1) (2005) 42–61.
[5] C.-J. Liao, C.-T. Tseng, P. Luarn, A discrete version of particle swarm optimization for flowshop scheduling problems, Comp. Oper. Res. 34 (10) (2007) 3099–3111.
[6] C. Blum, M. Sampels, An ant colony optimization algorithm for shop scheduling problems, J. Math. Modell. Algorithms 3 (3) (2004) 285–308.
[7] A. Hatamlou, Black hole: a new heuristic optimization approach for data clustering, Inf. Sci. 222 (2013) 175–184.
[8] M. Mahdavi, et al., Novel meta-heuristic algorithms for clustering web documents, Appl. Math. Comput. 201 (1) (2008) 441–451.
[9] P. Shelokar, V.K. Jayaraman, B.D. Kulkarni, An ant colony approach for clustering, Anal. Chim. Acta 509 (2) (2004) 187–195.
[10] S. Hashemi, et al., An image contrast enhancement method based on genetic algorithm, Pattern Recognit. Lett. 31 (13) (2010) 1816–1824.
[11] J. Bozek, et al., A survey of image processing algorithms in digital mammography, in: Recent Advances in Multimedia Signal Processing and Communications, Springer, 2009, pp. 631–657.
[12] B. Akay, D. Karaboga, A survey on the applications of artificial bee colony in signal, image, and video processing, Signal Image Video Process. 9 (4) (2015) 967–990.
[13] E. Alba, R. Martí, Metaheuristic Procedures for Training Neural Networks, vol. 35, Springer Science & Business Media, 2006.
[14] C. Blum, K. Socha, Training feed-forward neural networks with ant colony optimization: an application to pattern classification, Fifth International Conference on Hybrid Intelligent Systems (2005), IEEE.
[15] E. Valian, S. Mohanna, S. Tavakoli, Improved cuckoo search algorithm for feedforward neural network training, Int. J. Artif. Intell. Appl. 2 (3) (2011) 36–43.
[16] A. Lodi, S. Martello, D. Vigo, Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems, INFORMS J. Comput. 11 (4) (1999) 345–357.
[17] E. Hopper, B.C. Turton, A review of the application of meta-heuristic algorithms to 2D strip packing problems, Artif. Intell. Rev. 16 (4) (2001) 257–300.
[18] K. Fleszar, K.S. Hindi, New heuristics for one-dimensional bin-packing, Comp. Oper. Res. 29 (7) (2002) 821–839.
[19] K. Doerner, et al., Pareto ant colony optimization: a metaheuristic approach to multiobjective portfolio selection, Ann. Oper. Res. 131 (1–4) (2004) 79–99.
[20] Y. Crama, M. Schyns, Simulated annealing for complex portfolio selection problems, Eur. J. Oper. Res. 150 (3) (2003) 546–571.
[21] G. Di Tollo, A. Roli, Metaheuristics for the portfolio selection problem, Int. J. Oper. Res. 5 (1) (2008) 13–35.
[22] C.A.C. Coello, Evolutionary multi-objective optimization: basic concepts and some applications in pattern recognition, in: Pattern Recognition, Springer, 2011, pp. 22–33.
[23] M. Tupia, R. Cueva, J. Vicente, Double-relaxed GRASP algorithm for graphic pattern recognition in forensic odontology, Adv. Inf. Sci. Serv. Sci. 3 (9) (2011) 320–330.
[24] K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, Comp. Methods Appl. Mech. Eng. 194 (36) (2005) 3902–3933.
[25] J.H. Holland, Adaptation in Natural and Artificial Systems. An Introductory Analysis with Application to Biology, Control, and Artificial Intelligence, University of Michigan Press, Ann Arbor, MI, 1975.
[26] D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-wesley, Reading, MA, 1989, K. Ohno, K. Esfarjani, and Y Kawazoe: Computational Materi.
[27] T. Back, F.H.P. Hoffmeister Schwefel, A survey of evolution strategies, Proceedings of the 4th International Conference on Genetic Algorithms (1991).
[28] B. Javidy, A. Hatamlou, S. Mirjalili, Ions motion algorithm for solving optimization problems, Appl. Soft Comput. 32 (2015) 72–79.
[29] S. Mirjalili, A. Lewis, The Whale optimization algorithm, Adv. Eng. Softw. (95) (2016) 51–67.
[30] E.-G. Talbi, Metaheuristics: From Design to Implementation, vol. 74, John Wiley & Sons, 2009.
[31] J. Brownlee, Clever Algorithms: Nature-inspired Programming Recipes, Jason Brownlee, 2011.
[32] F.W. Glover, G.A. Kochenberger, Handbook of Metaheuristics, vol. 57, Springer Science & Business Media, 2006.
[33] C. Voudouris, E.P. Tsang, A. Alsheddy, Guided local search, in: Handbook of Metaheuristics, Springer, 2010, pp. 321–361.
[34] F.J. Solis, R.J.-B. Wets, Minimization by random search techniques, Math. Oper. Res. 6 (1) (1981) 19–30.
[35] N. Baba, T. Shoman, Y. Sawaragi, A modified convergence theorem for a random optimization method, Inf. Sci. 13 (2) (1977) 159–166.
[36] B. Mondal, K. Dasgupta, P. Dutta, Load balancing in cloud computing using stochastic hill climbing-a soft computing approach, Procedia Technol. 4 (2012) 783–789.
[37] M. Lozano, et al., Real-coded memetic algorithms with crossover hill-climbing, Evol. Comput. 12 (3) (2004) 273–302.
[38] H.R. Lourenço, O. Martin, T. Stützle, A beginner's introduction to iterated local search, Proceedings of MIC (2001).
[39] N. Mladenović, P. Hansen, Variable neighborhood search, Comp. Oper. Res. 24 (11) (1997) 1097–1100.
[40] E.K. Burke, G. Kendall, E. Soubeiga, A tabu-search hyperheuristic for timetabling and rostering, J. Heuristics 9 (6) (2003) 451–470.
[41] R. Battiti, G. Tecchiolli, The reactive tabu search, ORSA J. Comput. 6 (2) (1994) 126–140.
[42] B. Dogan, T. Olmez, A new metaheuristic for numerical function optimization: Vortex Search algorithm, Inf. Sci. 293 (2015) 125–145.
[43] A.A. Freitas, A survey of evolutionary algorithms for data mining and knowledge discovery, in: Advances in Evolutionary Computing, Springer, 2003, pp. 819–845.
[44] A. Kaveh, Advances in Metaheuristic Algorithms for Optimal Design of Structures, Springer, 2014.
[45] H.-G. Beyer, H.-P. Schwefel, Evolution strategies—a comprehensive introduction, Nat. Comput. 1 (1) (2002) 3–52.
[46] N. Hansen, A. Ostermeier, Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation, Proceedings of IEEE International Conference on Evolutionary Computation (1996), IEEE.
[47] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (2) (1999) 82–102.
[48] R. Storn, K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[49] X. Yao, Y. Liu, Fast evolutionary programming, Evolutionary Programming (1996), Citeseer.
[50] M. O'Neill, C. Ryan, Grammatical evolution, IEEE Trans. Evol. Comput. 5 (4) (2001) 349–358.
[51] C. Ferreira, Gene expression programming in problem solving, in: Soft Computing and Industry, Springer, 2002, pp. 635–653.
[52] O. Hasançebi, S.K. Azad, Adaptive dimensional search: a new metaheuristic algorithm for discrete truss sizing optimization, Comp. Struct. 154 (2015) 1–16.
[53] L. Cui, et al., Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations, Comp. Oper. Res. 67 (2016) 155–173.
[54] A.H. Gandomi, Interior search algorithm (ISA): a novel approach for global optimization, ISA Trans. 53 (4) (2014) 1168–1183.
[55] Z. Seif, M.B. Ahmadi, An opposition-based algorithm for function optimization, Eng. Appl. Artif. Intell. 37 (2015) 293–306.
[56] H. Salimi, Stochastic fractal search: a powerful metaheuristic algorithm, Knowledge-Based Syst. 75 (2015) 1–18.
[57] E. Aarts, J. Korst, W. Michiels, Simulated annealing, in: Search Methodologies, Springer, 2005, pp. 187–210.
[58] A. Kaveh, S. Talatahari, An enhanced charged system search for configuration optimization using the concept of fields of forces, Struct. Multidiscip. Optim. 43 (3) (2011) 339–351.
[59] A. Kaveh, S. Talatahari, A novel heuristic optimization method: charged system search, Acta Mech. 213 (3–4) (2010) 267–289.
[60] A. Kaveh, M. Khayatazad, A new meta-heuristic method: ray optimization, Comp. Struct. 112 (2012) 283–294.
[61] A.H. Kashan, A new metaheuristic for optimization: optics inspired optimization (OIO), Comp. Oper. Res. 55 (2015) 99–125.
[62] H. Shah-Hosseini, Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation, Int. J. Comput. Sci. Eng. 6 (1–2) (2011) 132–140.
[63] Ş.İ. Birbil, S.-C. Fang, An electromagnetism-like mechanism for global optimization, J. Global Optim. 25 (3) (2003) 263–282.
[64] O.K. Erol, I. Eksin, A new optimization method: big bang-big crunch, Adv. Eng. Softw. 37 (2) (2006) 106–111.
[65] A. Kaveh, V. Mahdavi, Colliding bodies optimization: a novel meta-heuristic method, Comp. Struct. 139 (2014) 18–27.
[66] Z.W. Geem, J.H. Kim, G. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[67] X. Jin, R.G. Reynolds, Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach, Proceedings of the 1999 Congress on Evolutionary Computation (1999), IEEE.
[68] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, 2007 IEEE Congress on Evolutionary Computation (2007).
[69] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (6) (2008) 702–713.
[70] Y.-J. Zheng, H.-F. Ling, J.-Y. Xue, Ecogeography-based optimization: enhancing biogeography-based optimization with ecogeographic barriers and differentiations, Comp. Oper. Res. 50 (2014) 115–127.
[71] T.K. Truong, K. Li, Y. Xu, Chemical reaction optimization with greedy strategy for the 0-1 knapsack problem, Appl. Soft Comput. 13 (4) (2013) 1774–1780.
[72] S.D. Müller, et al., Optimization based on bacterial chemotaxis, Trans. Evol. Comput. 6 (1) (2002) 16–29.
[73] H. Eskandar, et al., Water cycle algorithm—a novel metaheuristic optimization method for solving constrained engineering optimization problems, Comp. Struct. 110 (2012) 151–166.
[74] A.R. Mehrabian, C. Lucas, A novel numerical optimization algorithm inspired from weed colonization, Ecol. Inform. 1 (4) (2006) 355–366.
[75] X.-S. Yang, Flower pollination algorithm for global optimization, in: Unconventional Computation and Natural Computation, Springer, 2012, pp. 240–249.

[76] H. Shah-Hosseini, The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm, Int. J. Bio-Inspired Comput. 1 (1–2) (2009) 71–79.
[77] F. Merrikh-Bayat, The runner-root algorithm: a metaheuristic for solving unimodal and multimodal optimization problems inspired by runners and roots of plants in nature, Appl. Soft Comput. 33 (2015) 292–303.
[78] M. Ghaemi, M.-R. Feizi-Derakhshi, Forest optimization algorithm, Expert Syst. Appl. 41 (15) (2014) 6676–6687.
[79] M.S. Kiran, TSA: Tree-seed algorithm for continuous optimization, Expert Syst. Appl. 42 (19) (2015) 6686–6698.
[80] Y.-J. Zheng, Water wave optimization: a new nature-inspired metaheuristic, Comp. Oper. Res. 55 (2015) 1–11.
[81] R.V. Rao, V.J.D. Savsani Vakharia, Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems, Computer-Aided Des. 43 (3) (2011) 303–315.
[82] H. Shareef, A.A. Ibrahim, A.H. Mutlag, Lightning search algorithm, Appl. Soft Comput. 36 (2015) 315–333.
[83] A. Ahrari, A.A. Atai, Grenade explosion method—a novel tool for optimization of multimodal functions, Appl. Soft Comput. 10 (4) (2010) 1132–1140.
[84] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, New York, NY, 1995.
[85] M. Dorigo, V.A. Maniezzo Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 26 (1) (1996) 29–41.
[86] M. Dorigo, et al., Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, September 22–24, 2008, 2008, Proceedings. vol. 5217, Springer.
[87] M. Dorigo, T. Stützle, Ant Colony Optimization: Overview and Recent Advances. Handbook of Metaheuristics, 2010.
[88] X.S. Yang, Cuckoo Search via Levy Flights. Proceedings of World Congress on Nature & Biologically Inspired Computing (2009) 210–225.
[89] M. Dorigo, L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997) 53–66.
[90] D. Pham, et al., The bees algorithm—a novel tool for complex optimisation, in: Intelligent Production Machines and Systems-2nd I* PROMS Virtual International Conference, 3–14 July 2006, 2011, Elsevier.
[91] S. Farzi, Efficient job scheduling in grid computing with modified artificial fish swarm algorithm, Int. J. Comp. Theory Eng. 1 (1) (2009) 13.
[92] M.-Y. Cheng, D. Prayogo, Symbiotic organisms search: a new metaheuristic optimization algorithm, Comp. Struct. 139 (2014) 98–112.
[93] E. Cuevas, et al., A swarm optimization algorithm inspired in the behavior of the social-spider, Expert Syst. Appl. 40 (16) (2013) 6374–6384.
[94] M. Eusuff, K. Lansey, F. Pasha, Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization, Eng. Optim. 38 (2) (2006) 129–154.
[95] R.-q. Zhao, W.-s. Tang, Monkey algorithm for global numerical optimization, J. Uncertain Syst. 2 (3) (2008) 165–176.
[96] M. Hajiaghaei-Keshteli, M. Aminnayeri, Solving the integrated scheduling of production and rail transportation problem by Keshtel algorithm, Appl. Soft Comput. 25 (2014) 184–203.
[97] S.-C. Chu, P.-W. Tsai, J.-S. Pan, Cat swarm optimization, in: PRICAI 2006: Trends in Artificial Intelligence, Springer, 2006, pp. 854–858.
[98] A. Kaveh, N. Farhoudi, A new optimization method: dolphin echolocation, Adv. Eng. Softw. 59 (2013) 53–70.
[99] A.H. Gandomi, A.H. Alavi, Krill herd: a new bio-inspired optimization algorithm, Commun. Nonlin. Sci. Numer. Simulat. 17 (12) (2012) 4831–4845.
[100] S. Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm, Knowledge-Based Syst. 89 (2015) 228–249.
[101] G.-G. Wang, S. Deb, L.d.S. Coelho, Elephant herding optimization, in: 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI 2015), IEEE, Bali, Indonesia, 2015.
[102] S. Mirjalili, Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems, Neural Comput. Appl. (2015) 1–21.
[103] X.-S. Yang, Firefly algorithms for multimodal optimization, in: Stochastic Algorithms: Foundations and Applications, Springer, 2009, pp. 169–178.
[104] P. Oramus, Improvements to glowworm swarm optimization algorithm, Comp. Sci. 11 (2010) 7–20.
[105] A. Hatamlou, Heart: a novel optimization algorithm for cluster analysis, Prog. Artif. Intell. 2 (2–3) (2014) 167–173.
[106] D. Karaboga, C. Ozturk, A novel clustering approach: artificial bee colony (ABC) algorithm, Appl. Soft Comput. 11 (1) (2011) 652–657.
[107] M. Yazdani, F. Jolai, Lion optimization algorithm (LOA): a nature-inspired metaheuristic algorithm, J. Comput. Des. Eng. 3 (1) (2016) 24–36.
[108] N. Moosavian, B.K. Roodsari, Soccer league competition algorithm: a novel meta-heuristic algorithm for optimal design of water distribution networks, Swarm Evol. Comput. 17 (2014) 14–24.
[109] A.H. Kashan, League Championship Algorithm (LCA): An algorithm for global optimization inspired by sport championships, Appl. Soft Comput. 16 (2014) 171–200.
[110] D. Hiskey, February 9th: William G. Morgan invents a game called mintonette that is better known today as volleyball. 2012; Available from: http://www.todayifoundout.com/index.php/2012/02/february-9th-william-g-morgan-invents-a-game-called-mintonette-that-is-better-known-today-as-volleyball/.
[111] R.V. Rasmussen, M.A. Trick, Round robin scheduling—a survey, Eur. J. Oper. Res. 188 (3) (2008) 617–636.
[112] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowledge-Based Syst. 96 (2016) 120–133.
[113] S. Mirjalili, The ant lion optimizer, Adv. Eng. Softw. 83 (2015) 80–98.
[114] J.J.Q. Yu, V.O.K. Li, A social spider algorithm for global optimization, Appl. Soft Comput. 30 (2015) 614–627.
[115] C.A.C. Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Comp. Methods Appl. Mech. Eng. 191 (11) (2002) 1245–1287.
[116] C.A.C. Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2) (2000) 113–127.
[117] A.D. Belegundu, Study of mathematical programming methods for structural optimization, Diss. Abstr. Int. Part B: Sci. Eng. 43 (12) (1983) 1983.
[118] C.A.C. Coello, E.M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, Adv. Eng. Inf. 16 (3) (2002) 193–203.
[119] Q. He, L. Wang, An effective co-evolutionary particle swarm optimization for constrained engineering design problems, Eng. Appl. Artif. Intell. 20 (1) (2007) 89–99.
[120] E. Mezura-Montes, C.A.C. Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, Int. J. Gen. Syst. 37 (4) (2008) 443–473.
[121] A. Kaveh, S. Talatahari, An improved ant colony optimization for constrained engineering design problems, Eng. Comput. 27 (1) (2010) 155–182.
[122] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, App. Math. Comput. 188 (2) (2007) 1567–1579.
[123] L. Li, et al., A heuristic particle swarm optimizer for optimization of pin connected structures, Comp. Struct. 85 (7) (2007) 340–349.
[124] E. Sandgren, Nonlinear integer and discrete programming in mechanical design optimization, J. Mech. Des. 112 (2) (1990) 223–229.
[125] B. Kannan, S.N. Kramer, An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design, J. Mech. Des. 116 (2) (1994) 405–411.
[126] K. Deb, GeneAS: a robust optimal design technique for mechanical component design, in: Evolutionary Algorithms in Engineering Applications, Springer, 1997, pp. 497–514.
[127] A. Kaveh, S. Talatahari, Optimal design of skeletal structures via the charged system search algorithm, Struct. Multidiscip. Optim. 41 (6) (2010) 893–911.
[128] K. Ragsdell, D. Phillips, Optimal design of a class of welded structures using geometric programming, J. Eng. Ind. 98 (3) (1976) 1021–1025.
[129] K. Deb, Optimal design of a welded beam via genetic algorithms, AIAA J. 29 (11) (1991) 2013–2015.
[130] A. Kaveh, M. Motie Share, M. Moslehi, A new meta-heuristic algorithm for optimization: magnetic charged system search, Acta Mech. 224 (1) (2013) 85–107.