Neural Computing and Applications (2023) 35:9193–9223
https://doi.org/10.1007/s00521-022-08179-0

ORIGINAL ARTICLE

Velocity pausing particle swarm optimization: a novel variant for global optimization

Tareq M. Shami[1] · Seyedali Mirjalili[2,3] · Yasser Al-Eryani[4] · Khadija Daoudi[5] · Saadat Izadi[6] · Laith Abualigah[7,8,9,10,11]

Received: 7 June 2022 / Accepted: 8 December 2022 / Published online: 7 January 2023

Abstract

Particle swarm optimization (PSO) is one of the most well-regard metaheuristics with remarkable performance when solving diverse optimization problems. However, PSO faces two main problems that degrade its performance: slow convergence and local optima entrapment. In addition, the performance of this algorithm substantially degrades on high-dimensional problems. In the classical PSO, particles can move in each iteration with either slower or faster speed. This work proposes a novel idea called velocity pausing where particles in the proposed velocity pausing PSO (VPPSO) variant are supported by a third movement option that allows them to move with the same velocity as they did in the previous iteration. As a result, VPPSO has a higher potential to balance exploration and exploitation. To avoid the PSO premature convergence, VPPSO modifies the first term of the PSO velocity equation. In addition, the population of VPPSO is divided into two swarms to maintain diversity. The performance of VPPSO is validated on forty three benchmark functions and four real-world engineering problems. According to the Wilcoxon rank-sum and Friedman tests, VPPSO can significantly outperform seven prominent algorithms on most of the tested functions on both low- and high-dimensional cases. Due to its superior performance in solving complex high-dimensional problems, VPPSO can be applied to solve diverse real-world optimization problems. Moreover, the velocity pausing concept can be easily integrated with new or existing metaheuristic algorithms to enhance their performances. The Matlab code of VPPSO is available at: https://uk.mathworks.com/matlabcentral/fileexchange/119633-vppso.

Keywords Particle swarm optimization · PSO · Velocity pausing · Velocity pausing particle swarm optimization · VPPSO

1 Introduction

Optimization is an essential process that helps to achieve the best performance in many scientific fields such as engineering and artificial intelligence. As a consequence, the development of effective optimization algorithms is crucial. The need for such development has recently increased due to the increased difficulty level of optimization problems [1]. Although the traditional optimization approaches can be used to solve optimization problems, they have two main limitations: the requirement of gradient information that causes the conventional approaches to be unable to solve non-differentiable functions and local optima entrapment particularly when solving complex problems that have numerous local optima [2].

Metaheuristic algorithms are an effective way to solve diverse optimization problems regardless of their characteristics [3–5]. Due to its robustness, efficiency and simplicity, particle swarm optimization (PSO) has become one of the most widely used metaheuristic algorithms [6]. In addition, PSO has demonstrated superior performance when solving a wide range of optimization problems in various areas such as wireless communications [7, 8] and artificial intelligence [9, 10]. Other applications of PSO include truss layout [11], prestress design [12, 13], image segmentation [14] and flat-foldable origami tessellations [15]. Nonetheless, PSO still severely faces the problem of premature convergence [6, 16, 17]. Moreover, the

Extended author information available on the last page of the article performance of PSO in high-dimensional problems is poor [6]. This motivates the development of novel PSO variants that can overcome the limitations of the classical PSO algorithm and its state-of-the-art versions.

In PSO, the iterative process is split into two stages: exploration and exploitation. Exploration performs extensive search at the early stages of the search process in order to move toward the optimal solution [18]. It is essential that PSO algorithms have strong exploration abilities in order to escape from local optima entrapment. On the other hand, exploitation focuses on regions that have a great potential to be the place where the optimal solution can be found. Balancing between exploration and exploitation is crucial in order to be able to locate optimal solutions [19].

The no free lunch (NFL) theorem [20] states that an optimization algorithm that performs well on a given set of problems achieves poor performance when it is tested on a different class of problems. Many state-of-the-art PSO variants and metaheuristic algorithms have shown promising results on a certain class of optimization problems; nonetheless, they have shown degraded performance when they solve different sets of problems. This motivates the development of new PSO variants that can achieve the best solutions when they are applied to a diverse set of optimization problems.

This work proposes a novel PSO variant called velocity pausing particle swarm optimization (VPPSO). The main contributions of this work can be summarized as follows:

- A novel idea called velocity pausing is proposed where particles are provided with a third movement option (besides faster or slower speeds as in the classical PSO algorithm) that allows them to move with the same velocity as they did in the previous iteration.
- The proposed VPPSO algorithm modifies the first term of the classical PSO velocity equation to to avoid premature convergence.
- To maintain diversity, a two-swarm strategy is implemented where particles in the first swarm update their positions based on the classical PSO algorithm, whereas the remaining particles follow the global best position only to update their positions.
- A comprehensive comparison analysis that validates the effectiveness of VPPSO is carried out. The performance of VPPSO is evaluated on 23 classical benchmark functions, the CEC2019 test suite, the CEC2020 test functions and 4 real-world engineering problems. The performance of VPPSO on high-dimensional problems is also evaluated. VPPSO is compared with PSO, a recent high-performance PSO variant and five recent prominent metaheuristic algorithms.

The purpose of this work is to develop a high-performance robust PSO variant that can be used to optimize complex real-world problems. The rest of this work is organized as follows. Section 2 presents the related work that includes the classical PSO algorithm and its existing variants. In Sect. 3, the proposed VPPSO algorithm is described in detail. Section 4 presents the results of VPPSO and the competitive algorithms and it provides an in-depth discussion. The performance of VPPSO on real-world engineering problems is presented in Sect. 5. Section 6 concludes this work while Sect. 7 provides some potential research directions that can help to improve the PSO performance further.

2 Literature review

In this section, the preliminaries and essential definitions of PSO are first introduced. This includes the PSO source of inspiration and its mechanism. Although the original PSO algorithm has shown good optimization performance, it still faces some limitations such as local optima entrapment and slow convergence. This has motivated researchers to develop new PSO variants to tackle the aforementioned issues. Several related works on alleviating the PSO drawbacks are reviewed and discussed in the second subsection.

2.1 Particle swarm optimization

PSO is introduced by Kennedy and Eberhart [21] where its mechanism is inspired by social behaviors of birds flocking and fish schooling. In PSO, a swarm of particles flies in the search space to seek an optimal solution [22, 23]. Each particle i of the swarm in the D-dimensional space has a position and a velocity that can be mathematically written as follows:

$$\mathbf{V_i} = [V_{i1}, V_{i2}, ..., V_{iD}], \quad i = 1, 2, ..., N \tag{1}$$

$$\mathbf{X_i} = [X_{i1}, X_{i2}, ..., X_{iD}], \quad i = 1, 2, ..., N \tag{2}$$

where $\mathbf{V_i}$ and $\mathbf{X_i}$ are the velocity and position vectors of particle i, respectively, D is the number of dimensions and

N is the swarm size. At the beginning of the PSO optimization process, the velocity and position of each particle are randomly generated within specific ranges. During the PSO iterative process, a particle i is guided by the global best particle ($\mathbf{gbest} = [gbest_1, gbest_2, \ldots, gbest_D]$) which is the best particle that has been found so far and by its personal best position ($\mathbf{Pbest} = [Pbest_1, Pbest_2, \ldots, Pbest_D]$) to update its velocity and position, respectively, as follows:

$$V_{id}(t+1) = wV_{id}(t) + c_1 r_1 (Pbest_{id}(t) - X_{id}(t)) + c_2 r_2 (gbest_d(t) - X_{id}(t)) \tag{3}$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1) \tag{4}$$

where w is the inertia weight, c_1 and c_2 are the the cognitive and social acceleration coefficients, respectively, and r_1 and r_2 are two random variables distributed uniformly in the range [0,1]. The role of the inertia weight w is to avoid the velocity explosion problem faced by the standard PSO algorithm [21]. The acceleration coefficients c_1 and c_2 control the speed of a particle toward **Pbest** and **gbest**, respectively. These three PSO parameters (w, c_1 and c_2) play a crucial role for balancing the PSO exploration and exploitation abilities [24, 25]. Equation (3) is the core of the PSO algorithm, and it is the most essential formula that is needed to develop novel PSO variants.

After a particle updates its velocity and position, its personal best position is updated as follows:

$$Pbest_i(t+1) = \begin{cases} X_i(t+1) & \text{if } f(X_i(t+1)) < f(Pbest_i(t)) \\ Pbest_i(t) & \text{otherwise} \end{cases} \tag{5}$$

In Eq. (5), the personal best position of a particle i is updated only if the fitness of the newly generated particle X_i is better than the current fitness of $Pbest_i$. The next step in PSO is to update **gbest** based on the following:

$$gbest(t+1) = \begin{cases} Pbest_i(t+1) & \text{if } f(Pbest_i(t+1)) < f(gbest(t)) \\ gbest(t) & \text{otherwise} \end{cases} \tag{6}$$

The PSO process is repeated until a stopping criterion is satisfied.

2.2 Literature review of related works on PSO improvement

PSO has been modified by several strategies such as adjustment of PSO controlling parameters [26–28], multi-swarm schemes [29, 30], hybridization [31, 32] and new velocity updating mechanisms [33]. The controlling parameters of PSO, namely the inertia weight w, the cognitive component c_1 and the social component c_2 have a direct impact on the searching behavior of PSO [34]. Choosing the optimal values of w, c_1, c_2, is a challenging task since some values might perform well on certain optimization problems while the same values achieve poor performance on other sets of problems [6]. Many research efforts have attempted to develop new inertia weight strategies that aim to balance exploration and exploitation. One of the most well-known inertia weight approaches is time-varying inertia weight [35] that linearly decreases throughout the iterative process. In [35], the inertia weight is updated at each iteration as follows:

$$w(t) = (w_{max} - w_{\min})\left(\frac{T-t}{T}\right) + w_{\min} \tag{7}$$

where w_{max} and $w_{\min}$ represent the maximum and minimum values of the inertia weight, T is the maximum number of iterations while t is the current iteration. Other common inertia weight approaches that have been proposed to enhance the PSO performance are adaptive inertia weight [36–39], linearly decreasing inertia weight [27], nonlinear time-varying inertia weight [40–42], quadratic inertia weight [43], exponentially decreasing inertia weight [44, 45], chaotic inertia weight [46]. On the other hand, significant studies have attempted to improve the PSO performance by adjusting the PSO acceleration coefficients c_1 and c_2. The authors in [47] proposed a self-organizing hierarchical PSO where the two PSO acceleration coefficients vary with time (HPSO-TVAC). In HPSO-TVAC, c_1 and c_2 are initially assigned a large and small values, respectively, to enable strong exploration at the beginning of the PSO search process. Conversely, $c_1$and c_2 should have small and large values, respectively, at the final stages of the iterative process to allow particles exploit the search space significantly. The values of c_1 and c_2 are updated at each iteration as follows:

$$c_1 = \left(c_{1f} - c_{1i}\right)\frac{t}{T} + c_{1i} \tag{8}$$

$$c_2 = \left(c_{2f} - c_{2i}\right)\frac{t}{T} + c_{2i} \tag{9}$$

where the i and f subscripts represent the initial and final values, respectively. The authors in [48] proposed a fitness-based multi-role PSO (FMPSO) algorithm that adjusts its controlling parameters based on fitness. Similarly, a unique adaptive PSO (UAPSO) algorithm is developed in [25] to assign each particle unique inertia weight, c_1, and c_2 values based on its fitness. A phasor PSO (PPSO) algorithm is proposed in [49] where the first PSO velocity term that contains the inertia weight w is omitted, whereas c_1 and c_2 are replaced by phasor coefficients.

Multi-swarm techniques where particles are grouped into several sub-swarms based on a certain criterion have been widely used to enhance the PSO performance. In [50], a cooperative PSO (CPSO) approach is proposed where a number of swarms cooperate to optimize different segments of the solution vector. The work in [51] proposed a novel improved PSO algorithm based on individual difference evolution mechanism (IDE-PSO). According to particle's performances throughout the iterative process, particles are divided into several sub-swarms. The authors in [52] presented a new multi-swarm PSO algorithm based on dynamic learning strategy (PSO-DLS). In the proposed approach, particles are divided into conventional and communication particles where conventional particles perform exploitation while communication particles explore the search space. Using differential mutation operations, a two-swarm PSO algorithm is proposed in [53]. The authors in [54] proposed a multipopulation cooperative PSO (MPCPSO) algorithm that implements a difference mutation operator that can help to achieve better exploration. Another multi-swarm PSO variant is proposed in [55] where the total population is split into a main swarm and a hovering swarm. Utilizing an elite learning strategy, the authors in [56] presented a dynamic multi-swarm PSO (DMS-PSO-EL) algorithm.

One of the most common approaches in the field of metaheuristics that can help to enhance the performance is hybridization where the best properties of two algorithms are combined to develop a more efficient algorithm. In [31], a novel hybrid PSO with genetic algorithm (GA) is proposed where the mechanisms of PSO and the operators of GA (crossover and mutation) are implemented together to create a new generation of candidate solutions. The work in [57] hybridized PSO with Ant Colony Optimization (ACO). In the proposed approach, PSO and ACO execute their individual algorithms separately during the iterative process to create their own new solutions. However, the global best solution among the two algorithms is used to update the positions of particles and ants at each iteration. PSO has been also hybridized with other optimization algorithms such as simulated annealing (SA) [58], gray wolf optimization (GWO) [59], firefly algorithm (FA) [60] and whale optimization algorithm (WOA) [61] where in all proposed approaches the hybrid PSO versions outperform the individual PSO algorithm.

Besides the three aforementioned strategies, many works have proposed other methods such as implementation of different neighbourhood structures and development of new velocity updating mechanisms to enhance the PSO performance. In [62], a Fully Informed PSO (FIPS) algorithm is developed where a particle requires the positions information of its neighbors to update its velocity. A new PSO algorithm is developed in [63] by proposing a dynamic PSO neighbourhood strategy that continuously updates the neighbourhood of each particle throughout the iterative process. The four PSO search strategies presented in Comprehensive Learning PSO (CLPSO) [64], Unified PSO (UPSO) [65], Linearly Decreasing Inertia Weight PSO (LDWPSO) [35], distance-based locally informed PSO (LISP) [66] are combined into one algorithm to develop a PSO with Strategy Dynamics (SDPSO) algorithm [67]. The authors in [68] have proposed an enhanced social learning PSO algorithm that updates the best three particles based on a differential mutation approach. To solve constrained optimization problems, a novel PSO variant named PSO+ is proposed in [69] where the authors have proposed a novel strategy to update the positions of particles. A new PSO variant called Generalized PSO (GEPSO) is introduced in [33] where the velocity of the classical PSO algorithm is modified by including two new terms. A novel chaotic grouping PSO algorithm that implements a Dynamic Regrouping Strategy (CGPSO-DRS) is proposed in [70]. The work in [71] has developed an enhanced PSO algorithm by using complex-order derivatives. In [72], a new PSO variant is developed by applying two strategies: multi-exemplar and forgetting ability. Some recent prominent PSO variants are presented in Table 1. The PSO variants mentioned in this section can be applied to optimize various problems including truss layout [11], image segmentation [14], wireless communications [7], prestress design [12, 13] and flat-foldable origami tessellations [15]. Although existing PSO variants have shown that they can significantly improve the performance of the classical PSO algorithm, the effectiveness of [33, 48, 49, 52–54, 56, 63, 68, 71, 72] on real-world optimization problems is not validated. In addition, the performances of [33, 48, 52, 53, 56, 69, 71] on high-dimensional problems are not investigated. In [33, 48, 54–56, 68, 71], the proposed algorithms are compared with PSO variants only without comparing their performances with other well-known metaheuristics such as GWO and WOA. Finally, the works in [48, 53–56, 68, 70] require massive number of function evaluations to achieve competitive results.

Table 1 Some recent prominent PSO variants

Algorithm	Contribution(s)	High-dimensional problems	Benchmark functions	Statistical test(s)	Engineering problems	FEs
Two-swarm learning PSO (TSLPSO) [73]	Dimensional learning and comprehensive learning strategies	No	16 functions and CEC2014	Yes	Yes	3×10^5
PSO-ALS [74]	An adaptive learning strategy	No	15 functions and CEC2017	Yes	Yes	2×10^5
Expanded PSO (XPSO) [72]	Multiple exemplars and forgetting ability	No	CEC2013	Yes	No	$10000D$
Triple archives PSO (TAPSO) [75]	A three archives strategy	No	30 classical functions	Yes	Yes	$10000D$
Novel social learning PSO (NSLPSO) [68]	A new social learning strategy	No	CEC2013	Yes	No	$10000D$
Pyramid PSO [76]	Novel cooperation and competition strategies	No	CEC2013 and CEC2017	Yes	No	$10000D$
Multi-population cooperative PSO (MPCPSO) [54]	Multi-dimensional comprehensive learning approach	Yes	16 classical functions	No	No	200,000
Bee-foraging learning PSO (BFL-PSO) [77]	Integration of PSO and artificial bee colony algorithm	No	CEC2014	Yes	Yes	$10000D$
Generalized PSO (GEPSO) [33]	Modification of the velocity equation	No	16 classical functions	No	No	10,000
Adaptive strategy PSO (ASPSO) [78]	PSO is hybridized with an adaptive strategy	No	CEC2017	Yes	Yes	50,000
PSO+ [69]	A new particles update strategy	No	24 classical functions	Yes	Yes	30,000

FEs denotes the number of function evaluations

3 Velocity pausing particle swarm optimization

This work proposes a novel idea called velocity pausing where each particle does not have to update its velocity at each iteration. In other words, a particle is allowed to move with the same velocity as it did in the previous iteration. This idea allows particles to have the potential of moving with three different speeds, i.e., slower speed, faster speed and constant speed unlike the standard PSO algorithm where particles move with only faster speed or slower speed. The main advantage of velocity pausing is the addition of a third movement option (constant speed) that can help to balance exploration and exploitation and avoid the severe premature convergence of the classical PSO. The velocity pausing concept can be written mathematically as follows:

$$V_i(t+1) = \begin{cases} V_i(t) & \text{if } rand < \alpha \\ wV_i(t) & \text{Otherwise} \\ +c_1 r_3(Pbest_i(t) - X_i(t)) & \\ +c_2 r_4(gbest(t) - X_i(t)) & \end{cases} \tag{10}$$

where $V_i(t)$ and $V_i(t+1)$ are the velocities of particle i at iterations t and $t+1$, respectively, α is the velocity pausing parameter. In case the pausing parameter α has a value higher than 1, all particles will update their velocities at each iteration exactly in the same way as the classical PSO algorithm does. This situation is undesired since no velocity pausing can occur. On the other hand, an extremely low value of α will force particles to move with constant speed and it will restrict them from moving with faster or slower speed. Therefore, it is crucial to choose the best α value to achieve a balanced velocity pausing scenario that can lead to an optimal performance. To further help PSO avoid premature convergence, the velocity equation of the conventional PSO algorithm is modified by changing the first velocity term and omitting its inertia weight component as follows:

$$V_i(t+1) = V_i(t)^{r_5 a(t)} + c_1 r_6 (Pbest_i(t) - X_i(t)) + c_2 r_7 (gbest(t) - X_i(t)) \tag{11}$$

where $a(t)$ is mathematically written as follows:

$$a(t) = \exp^{-\left(\frac{bt}{T}\right)^b} \tag{12}$$

In Eq. 12, b is constant. By applying the velocity pausing concept and utilizing the modified velocity equation in (11), a particle in VPPSO updates its velocity as follows:

$$V_i(t+1) = \begin{cases} V_i(t) & \text{if } rand < \alpha \\ V_i(t+1) \quad \text{as in (11)} & \text{Otherwise} \end{cases} \tag{13}$$

Utilizing Equation (13), the position of a particle i is updated as follows:

$$X_i(t+1) = X_i(t) + V_i(t+1) \tag{14}$$

To maintain diversity and avoid premature convergence, the proposed algorithm divides the total population N into two swarms. The first swarm consists of N_1 particles that update their velocities and positions based on the classical PSO mechanism except the following: The first term of the velocity equation is modified and the velocity pausing concept is applied as shown in Eq. 13. The second swarm has N_2 particles that rely only on **gbest** to update their positions. Each particle in the second swarm updates its position as follows:

$$X_i(t+1) = \begin{cases} gbest + a(t) r_8 |gbest|^{a(t)} & \text{if } r_9 < 0.5 \\ gbest - a(t) r_{10} |gbest|^{a(t)} & \text{Otherwise} \end{cases} \tag{15}$$

The optimization process of VPPSO starts by randomly generating the velocities and positions of all particles. During the VPPSO iterative process, particles in the first swarm update their velocities and positions based on Eqs. 13 and 14, respectively, while particles in the second swarm update their positions based on (15). The next step of VPPSO is to evaluate the fitness of all particles. Considering the first swarm, the personal best positions of particles are updated based on Eq. 5 followed by updating the global best position based on (6). The global best position is also updated in the second swarm of VPPSO if a particle in the second swarm can achieve a better fitness. The VPPSO process is repeated until a stopping criterion is satisfied. The Pseudo-code of VPPSO is provided in Algorithm 1. Applying Algorithm 1 is important to solve complex real-world problems particularly high-dimensional problems. Moreover, Algorithm 1 includes velocity pausing, a new velocity equation and a two-swarm strategy that can better balance exploration and exploitation and enhance diversity.

The flowchart of the proposed VPPSO algorithm is presented in Fig. 1. The modifications of VPPSO are highlighted in green colour. The flowchart shows the first VPPSO modification which is updating the velocities of PSO particles based on a new proposed equation. The new velocity equation changes the first term of the original PSO velocity equation to avoid premature convergence. Moreover, the proposed velocity equation implements velocity pausing to help balancing exploration and exploitation. The other modification of VPPSO is the addition of a second swarm where particles in this swarm update their positions differently. The VPPSO two-swarm strategy is needed to enhance diversity. For PSO, VPPSO and the other existing metaheuristic algorithms, the **gbest** vector is entirely replaced at iteration t if its fitness is better than the fitness of **gbest** at iteration $t-1$. This is not the optimal approach for **gbest** replacement as some dimensions of **gbest** at iteration t may be not better than their corresponding dimensions at iteration $t-1$. This **gbest** replacement problem has been tackled in [50]; however, the proposed approach is computationally prohibitive. Other novel approaches are needed to replace the **gbest** vector more efficiently.

3.1 Complexity analysis

The complexity of swarm algorithms is mainly dependant on the population size N, number of dimensions D, the cost of function evaluations C and the maximum number of function evaluations. Functions are evaluated N times at each iteration t; thus, the number of the overall function evaluations is NT where T is the maximum number of iterations. In PSO and other swarm algorithms, the complexity can be divided into two parts: initialization and the iterative loop. The initialization phase randomly generates particles and evaluates their fitness. Generating random particles and evaluating their fitness have complexities of $O(ND)$ and $O(NC)$. As a result, the initialization complexity of PSO becomes $O(ND + NC)$. The PSO iterative loop consists of positions update, function evaluations and memory savings

Algorithm 1 Pseudo-code of VPPSO

```
Define the values of N, N_1, N_2, α, T, and a and set f(gbest) = ∞
for i = 1 : N do
    Randomly generate the position of the particle i (X_i) and set its velocity to zero V_i = 0.
    Evaluate the fitness of particle i, i.e., (f(X_i))
    Set Pbest_i = X_i and f(Pbest_i) = f(X_i)
    if f(Pbest_i) < f(gbest) then
        gbest = Pbest_i
        f(gbest) = f(Pbest_i)
    end if
end for
for t = 1 : T do
    for i = 1 : N do
        if i ≤ N_1 then
            Update the particle's velocity V_i and position X_i based on Equations 13 and 14, respectively
        else
            Update the particle's position X_i based on Equation 15
        end if
    end for
    for i = 1 : N do
        Evaluate the fitness of particle i, i.e., f(X_i)
        if i ≤ N_1 then
            if f(X_i) < f(Pbest_i) then
                Pbest_i = X_i
                f(Pbest_i) = f(X_i)
                if f(Pbest_i) < f(gbest) then
                    gbest = Pbest_i
                    f(gbest) = f(Pbest_i)
                end if
            end if
        else
            if f(X_i) < f(gbest) then
                gbest = X_i
                f(gbest) = f(X_i)
            end if
        end if
    end for
end for
return gbest
```

where their computational complexities are given as $O(TND)$, $O(TNC)$ and $O(TN)$, respectively. The overall PSO complexity can be written as follows:

$$O(PSO) = O(ND + NC + TND + TNC + TN) \tag{16}$$

The initialization complexity of VPPSO is the same as PSO which is given as $O(ND + NC)$. In the iterative loop of VPPSO, the complexity is the same as PSO except that the the VPPSO second swarm does not involve memory savings.

The overall VPPSO complexity can be written as follows:

$$O(VPPSO) = O(ND + NC + TND + TNC + TN_1) \tag{17}$$

From (17), it is clear that VPPSO modifies the original PSO algorithm without increasing its complexity. On the contrary, the VPPSO complexity is lower than the complexity of the standard PSO version as the second swarm of VPPSO does not require the information of the personal best positions as in the original PSO. The complexity of VPPSO can be further reduced by relying less on the personal best positions of PSO as they require memory savings and by the implementation of new low-complex searching strategies. In case $N_1 = N_2$ as in this work, $N_1 = \frac{N}{2}$ which slightly reduces the complexity of VPPSO to:

$$O(VPPSO) = O\left(ND + NC + TND + TNC + T\frac{N}{2}\right) \tag{18}$$

4 Results and discussion

The effectiveness of VPPSO is first validated by testing it on twenty-three classical benchmark functions that have been widely used to evaluate the performance of new metaheuristic algorithms or their variants [79–83]. These conventional functions are grouped into three categories: unimodal functions (Table 2), multimodal functions (Table 3) and multimodal functions with fixed dimensions (Table 5). The mathematical expressions of the twenty-three functions are shown in Tables 2, 3 and 4. In addition, these three tables show the search range of each benchmarking function as well as its optimal value. The main purpose of testing novel metaheuristic algorithms on unimodal functions (f_1-f_7) is to assess their exploitation performance since a unimodal function possesses only a single optima. On the other hand, multimodal functions (f_8–f_{23}) help to evaluate the exploration ability of an optimization algorithm as they have multiple optima. The main distinction between the f_8–f_{13} and f_{14}–f_{23} multimodal functions is that the dimensions of f_8–f_{13} can be varied while

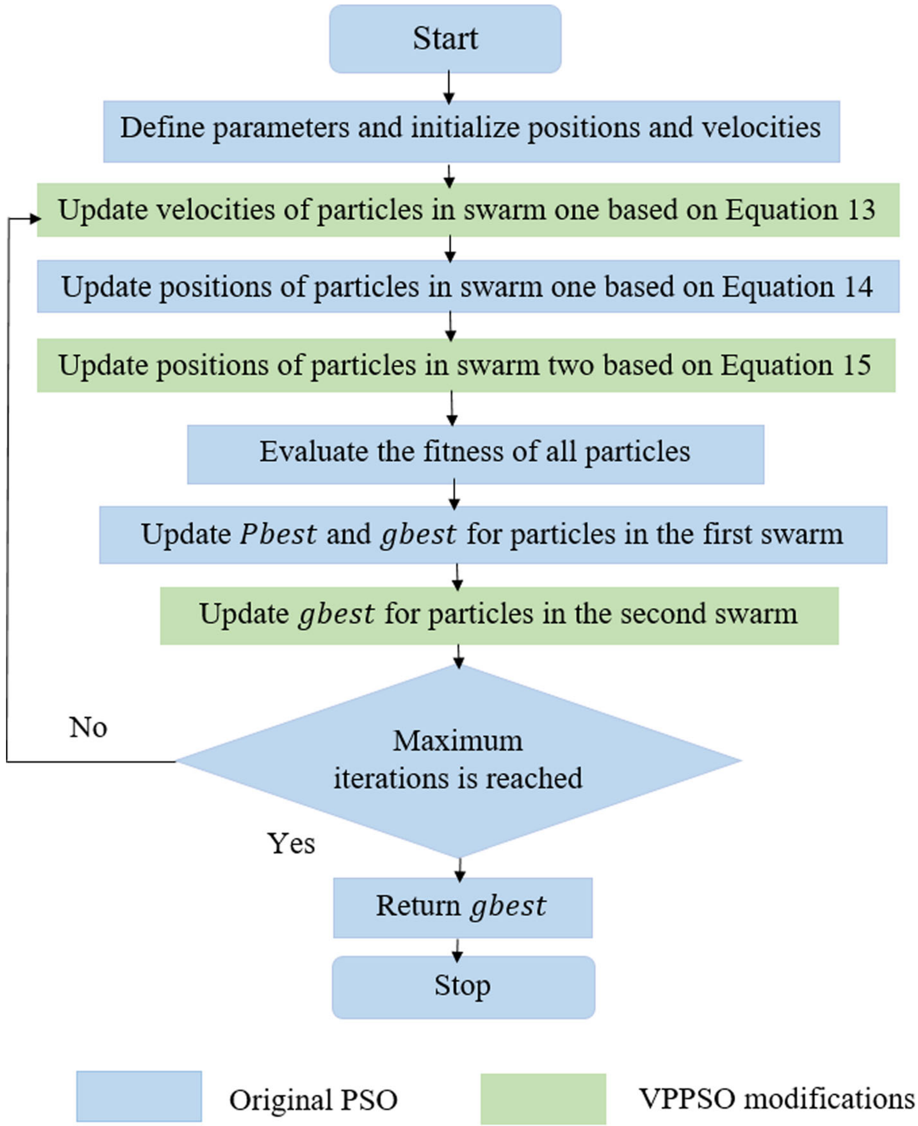

Fig. 1 Flowchart of the VPPSO algorithm

Table 2 Unimodal test functions

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

f_{14}–f_{23} have fixed dimensions. Moreover, the search range of f_8–f_{13} and f_{14}–f_{23} are different. The performance of the proposed VPPSO algorithm is further validated by testing it on the CEC2019 test suite that consists of ten benchmark functions. Table 5 lists the names, search ranges, dimensions and optimal values of the CEC2019 functions. To further challenge VPPSO, VPPSO is applied to solve the ten CEC2020 complex optimization problems. As shown in Table 6, the CEC2020 test suite consists of one unimodal function (f_{34}), three basic functions ($f_{35} - f_{37}$), three hybrid functions ($f_{38} - f_{40}$) and three composition functions ($f_{41} - f_{43}$). A summary of the CEC2020 functions that include their names, search range and optimal values is shown in Table 6. VPPSO is compared with the classical PSO algorithm as well as with a recent high-performance PSO variant known as PPSO [49]. PPSO has shown that it outperforms several existing well-known PSO variants including CLPSO [64], adaptive particle swarm optimization (APSO) [39] and FIPS [62]. Besides the PSO algorithms, the performance of VPPSO is compared with five

Table 3 Multimodal test functions

Function	Range	$f_{\min}$
$f_8(x) = \sum_{i=1}^{n} -x_i sin(\sqrt{\mid x_i \mid})$	[−500,500]	$-418.9829 \times Dim$
$f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	[−5.12,5.12]	0
$f_{10}(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + e$	[−32,30]	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[−600,600]	0
$f_{12}(x) = \frac{\pi}{n}\{10sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10sin^2(\pi y_i + 1)\right] + (y_n - 1)^2 + \sum_{i=1}^{n} u(x_i, 10, 100, 4)\}$	[−50,50]	0
$y_i = 1 + \frac{x_i + 1}{4}$		
$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$		
$f_{13}(x) = 0.1\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	[−50,50]	0

Table 4 Fixed-dimension multimodal test functions

Function	Dim	Range	$f_{\min}$
$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	2	[−65,65]	1
$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	[−5,5]	0.00030
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[−5,5]	−1.0316
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosx_1 + 10$	2	[−5,5]	0.398
$f_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	2	[−2,2]	3
$f_{19}(x) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3	[1,3]	−3.86
$f_{20}(x) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	[0,1]	−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

Table 5 CEC2019 test functions

No	Function name	Dim	Range	f_{min}
f_{24}	Storn's Chebyshev Polynomial Fitting Problem	9	[−8192,8192]	1
f_{25}	Inverse Hilbert Matrix Problem	16	[−16,384,16,384]	1
f_{26}	Lennard–Jones Minimum Energy Cluster	18	[−4,4]	1
f_{27}	Rastrigin's Function	10	[−100,100]	1
f_{28}	Griewangk's Function	10	[−100,100]	1
f_{29}	Weierstrass Function	10	[−100,100]	1
f_{30}	Modified Schwefel's Function	10	[−100,100]	1
f_{31}	Expanded Schaffer's F6 Function	10	[−100,100]	1
f_{32}	Happy Cat Function	10	[−100,100]	1
f_{33}	Ackley Function	10	[−100,100]	1

Table 6 CEC2020 test functions

No	Function name	Range	f_{min}
f_{34}	Shifted and Rotated Bent Cigar Function	[−100,100]	100
f_{35}	Shifted and Rotated Schwefel's Function	[−100,100]	1100
f_{36}	Shifted and Rotated Lunacek biRastrigin Function	[−100,100]	700
f_{37}	Expanded Rosenbrock's plus Griewangk's Function	[−100,100]	1900
f_{38}	Hybrid Function 1 ($N = 3$)	[−100,100]	1700
f_{39}	Hybrid Function 2 ($N = 4$)	[−100,100]	1600
f_{40}	Hybrid Function 3 ($N = 5$)	[−100,100]	2100
f_{41}	Composition Function 1 ($N = 3$)	[−100,100]	2200
f_{42}	Composition Function 2 ($N = 4$)	[−100,100]	2400
f_{43}	Composition Function 3 ($N = 5$)	[−100,100]	2500

Table 7 Parameter settings of all compared algorithms

Algorithm	Parameter	Value
VPPSO	α, N_1, N_2	0.3, 15, 15
PSO	C_1, C_2, w	2, 2, 0.9–0.4
PPSO		
HGSO	Cluster size, M_1, M_2,	5, 0.1 , 0.2
	K, α, β	1, 1, 1
GWO	a	2–0
SSA	Position update probability	0.5
WOA	a	2–0
AOA	C_1, C_2	2, 6

prominent recent metaheuristic algorithms: GWO [79], Henry gas solubility optimization (HGSO) [84], salp swarm algorithm (SSA) [85], WOA [81] and Archimedes optimization algorithm (AOA) [86]. The results of these five algorithms have shown superior optimization performance when compared with many optimization algorithms including equilibrium optimizer (EO) [2], sine-cosine algorithm (SCA) [87], L-SHADE, GA, gravitational search algorithm (GSA) [88] and differential evolution (DE) [89]. For all algorithms, results are averaged over 30 independent runs while the population size is 30. Following the recommendations of the original references, the parameter settings of all compared algorithms are summarized in Table 7.

4.1 Exploitation analysis

To evaluate the exploitation ability of the proposed approach, its performance is compared with seven algorithms on seven unimodal functions (f_1–f_7). The statistical results of unimodal functions including the average fitness and standard deviation are recorded in Table 8. From Table 8, it is clear that VPPSO outperforms all other algorithms on all seven functions except f_7. VPPSO achieves competitive results on f_7 that allows it to be ranked second. It can be also noted that VPPSO is the only algorithm that can achieve the optimal solutions for f_1, f_2, f_3

and f_4. From Table 8, it is evident that VPPSO can achieve a near optimal solution when solving f_5 while all other algorithms achieve poor performances when solving the same function. Overall, VPPSO is ranked first according to the Friedman test as can be seen in Table 8. These results have shown that VPPSO possesses a robust exploitation abilities.

4.2 Exploration analysis

The exploration performance of VPPSO is evaluated on 16 multimodal functions (f_8–f_{23}) that consist of functions with different dimensional sizes and different search ranges as illustrated in Tables 2 and 3. The statistical results of all algorithms for the 16 multimodal functions are provided in Table 8 (f_8–f_{13}) and Table 9 (f_{14}–f_{23}) when $D = 30$. From these two tables, it is clear that VPPSO obtains better solutions on $f_8, f_{12}, f_{13}, f_{20}, f_{21}, f_{22}$ and f_{23} compared with other algorithms. The two tables also show that VPPSO achieves the best solutions equally with a few other algorithms on $f_9, f_{11}, f_{16}, f_{17}$ and f_{18}. It is also remarkable that VPPSO can achieve the optimal solutions for 8 functions, i.e., $f_9, f_{11}, f_{16}, f_{17}, f_{18}, f_{21}, f_{22}$ and f_{23}. For f_{10}, f_{14}, f_{15} and f_{19}, VPPSO shows a competitive performance compared with all other algorithms. According to Friedman mean rank, the proposed VPPSO approach is ranked first when solving the multimodal functions. This demonstrates the strong exploration ability of VPPSO.

4.3 Impact of high dimensionality

One of the main problems of PSO is its poor performance on high-dimensional problems. Therefore, it is crucial to develop a novel PSO variant that can achieve effective and consistent performance on low- and high-dimensional optimization problems. The performance of VPPSO on high-dimensional cases is investigated by increasing the number of dimensions of functions $f_1 - f_{13}$ to 100 and 500. Tables 10 and 11 show the comparative results for all algorithms on $f_1 - f_{13}$ functions when $D = 100$ and $D = 500$, respectively. As Table 8 ($D = 30$), Table 10 ($D = 100$) and Table 11 ($D = 500$) show, VPPSO achieves a consistent performance on the tested functions unlike other algorithms. It is also notable from Tables 10 and 11 that VPPSO still achieves the optimal solutions for $f_1 - f_4$, f_9 and f_{11} when $D = 100$ and $D = 500$, respectively. Tables 8, 10 and 11 demonstrate that all other algorithms particularly PSO and SSA achieved degraded performance as the number of dimensions increases. Overall, according to the Friedman mean rank, VPPSO achieves the best high-dimensional performance in comparison with the seven other algorithms as Tables 10 and 11 show.

4.4 Performance of VPPSO on the CEC2019 and CEC2020 test functions

The performance of VPPSO on the CEC2019 test functions is recorded in Table 12. From Table 12, it is clear that VPPSO outperforms all algorithms on 7 functions out of 10. Table 12 also shows that VPPSO and HGSO are able to obtain the optimal solution of f_{24} while the remaining algorithms achieve poor performance. For f_{27} and f_{29}, VPPSO achieves the second best solutions while the best solutions are achieved by GWO. Based on the Friedman mean rank, VPPSO achieves the best performance as Table 12 illustrates.

The 10 CEC2020 complex optimization problems are used to further challenge the performance of VPPSO. Table 13 presents a performance comparison of VPPSO and other algorithms when they are applied to solve the CEC2020 test functions. As Table 13 shows, it can be seen that VPPSO can perform better than all compared algorithms on $f_{34}, f_{35}, f_{36}, f_{39}$ and f_{43} while its performance on f_{37} is equal to the performances of all other algorithms. For the remaining functions, the performance of VPPSO is comparable to other algorithms. The results in Table 13 demonstrates the strength and superiority of VPPSO to solve complex optimization problems. The Friedman mean rank presented in Table 13 shows that VPPSO achieves the first rank when compared with the 7 well-known and high-performance optimization algorithms.

4.5 Sensitivity analysis

This subsection investigates the impact of the VPPSO parameters on its optimization performance. The main parameter of VPPSO that is expected to have a direct and significant influence of the VPPSO behavior is the velocity pausing parameter α where α can have any value that is equal to or less than one. A value of $\alpha = 1$ represents the classical PSO algorithm. To study the impact of α on the performance of VPPSO, ten different scenarios are studied where α varies from 1 to 0.1 in steps of 0.1. Another main parameter that can affect the performance of VPPSO is the number of particles per swarm as VPPSO is a two-swarm algorithm. Three different swarm-size cases are studied where the size of the PSO swarm and the size of the second swarm are $N_1 = 20$, $N_2 = 10$, and $N_1 = 15$, $N_2 = 15$, and $N_1 = 10$, $N_2 = 20$, respectively. For each swarm-size case, results are generated for the 23 classical benchmark functions ($f_1 - f_{23}$) while considering the 10 different scenarios of α. Tables 14, 15 and 16 present the results of the average fitness and the standard deviation for swarm-size case 1, swarm-size case 2 and swarm-size case 3, respectively,

Table 8 Statistical results of f_1-f_{13} when $D = 30$

Fun		VPPSO	PSO	PPSO	HGSO	GWO	SSA	WOA	AOA
f_1	Mean	0	2.03E−03	2.35E−02	5.37E−185	8.15E−28	1.59E−07	6.38E−70	2.18E−84
	Std	0	4.62E−03	2.73E−02	0	1.23E−27	2.56E−07	3.48E−69	1.14E−83
f_2	Mean	0	1.27E−02	1.04E−01	1.00E−90	8.98E−17	2.00E+00	2.42E−51	5.13E−47
	Std	0	1.34E−02	7.25E−02	4.01E−90	7.05E−17	1.84E+00	8.44E−51	2.80E−46
f_3	Mean	0	4.35E+02	2.04E−01	1.11E−134	1.21E−05	1.49E+03	4.28E+04	2.80E−64
	Std	0	1.73E+02	2.69E−01	6.10E−134	2.50E−05	1.05E+03	1.60E+04	1.53E−63
f_4	Mean	0	3.36E+00	2.59E−02	2.69E−83	9.62E−07	1.22E+01	4.74E+01	5.57E−40
	Std	0	7.03E−01	4.04E−02	1.47E−82	1.36E−06	3.88E+00	3.15E+01	3.05E−39
f_5	Mean	1.29E−03	7.47E+01	2.89E+01	2.84E+01	2.71E+01	2.69E+02	2.80E+01	2.88E+01
	Std	1.52E−03	5.59E+01	5.03E−01	4.08E−01	7.98E−01	4.47E+02	4.30E−01	9.83E−02
f_6	Mean	1.20E−07	2.51E−03	3.63E−01	4.22E+00	7.47E−01	1.47E−07	3.65E−01	5.69E+00
	Std	3.63E−08	7.43E−03	2.39E−01	5.50E−01	3.54E−01	1.59E−07	2.16E−01	3.67E−01
f_7	Mean	6.10E−04	2.08E−02	2.73E−03	2.00E−04	1.76E−03	1.60E−01	4.32E−03	7.69E−04
	Std	5.95E−04	5.11E−03	2.22E−03	1.36E−04	6.97E−04	4.22E−02	4.76E−03	5.49E−04
f_8	Mean	−1.22E+04	−6.70E+03	−9.62E+03	−6.04E+03	−6.02E+03	−7.45E+03	−1.00E+04	−3.58E+03
	Std	5.00E+02	6.91E+02	1.43E+03	4.35E+03	7.92E+02	7.78E+02	1.72E+03	2.37E+02
f_9	Mean	0	5.26E+01	4.31E−01	0	3.76E+00	5.75E+01	0	0
	Std	0	1.69E+01	1.83E+00	0	5.22E+00	1.86E+01	0	0
f_{10}	Mean	7.99E−15	3.46E−01	6.06E−02	1.00E−15	9.96E−14	2.57E+00	4.32E−15	2.54E−15
	Std	0	5.70E−01	1.17E−01	6.48E−16	1.69E−14	6.13E−01	2.37E−15	1.80E−15
f_{11}	Mean	0	1.49E−02	8.76E−02	0	3.09E−03	1.66E−02	1.96E−02	1.90E−02
	Std	0	1.66E−02	9.34E−02	0	6.57E−03	1.37E−02	6.13E−02	1.04E−01
f_{12}	Mean	1.83E−07	3.45E−02	3.67E−02	4.42E−01	4.21E−02	8.00E+00	2.45E−02	8.19E−01
	Std	3.52E−07	4.96E−02	5.72E−02	1.18E−01	1.98E−02	3.25E+00	2.38E−02	1.77E−01
f_{13}	Mean	1.83E−03	8.95E−03	1.64E+00	2.79E+00	6.46E−01	1.56E+01	5.17E−01	2.91E+00
	Std	4.16E−03	1.29E−02	5.06E−01	1.31E−01	2.18E−01	1.31E+01	2.27E−01	5.19E−02
Mean rank		1.30	5.07	5.15	3.15	4.30	6.38	4.15	4.53
Rank		1	6	7	2	4	8	3	5

where in each swarm-size case α is varied from 1 to 0.1. From these three tables, it is evident that a better performance is achieved when α decreases from 1 to 0.3 while the performance starts to degrade when the value of α is less than 0.3. It is also clear from the overall rank that the best performance is achieved when $\alpha = 0.3$ in all swarm-size cases. For any value α, it is observed from Tables 14, 15 and 16 that swarm-size case 2 where $N_1 = 15$ and $N_2 = 15$ outperforms both swarm-size case 1 and swarm-size case 3. Overall, the best performance is achieved when $\alpha = 0.3$, $N_1 = 15$ and $N_2 = 15$.

4.6 Convergence analysis

Convergence to local optima is a major challenge faced by most of metaheuristic algorithms including PSO. To tackle this issue, it is crucial to achieve a proper balance between exploration and exploitation. PSO has shown that it can be easily trapped in local optima resulting in a poor solution accuracy [6, 16]. The convergence curves of VPPSO, PSO and the best four algorithms (according to Friedman test as shown later in Table 17), i.e., HGSO, PPSO, GWO and AOA, are presented in Fig. 2. One of the main limitations of PSO is that particles prematurely converge toward a local solution. This problem can be clearly seen from Fig. 2e where PSO prematurely converges toward a sub-optimal value at the 77^{th} iteration. It is evident that the PSO particles cannot make any further improvements from the 77^{th} iteration until the end of the PSO searching process. This happens because of the poor exploration ability of the PSO algorithm when the algorithm is trapped in a local optima. On the other hand, Fig. 2e shows that VPPSO can avoid premature convergence by performing efficient exploration that can help to find better solutions as the number of iterations increase. Figure 2f shows another example where PSO suffers from the premature

Table 9 Statistical results of f_{14}-f_{23}

Fun		VPPSO	PSO	PPSO	HGSO	GWO	SSA	WOA	AOA
f_{14}	Mean	1.13E+00	3.00E+00	9.98E−01	1.94E+00	4.87E+00	1.163E+00	3.61E+00	1.04E+00
	Std	3.29E−01	2.59E+00	2.16E−16	7.54E−01	4.44E+00	5.26E−01	3.78E+00	1.81E−01
f_{15}	Mean	1.15E−03	2.51E−03	4.47E−03	3.90E−04	5.14E−03	1.52E−03	6.54E−04	8.54E−04
	Std	3.63E−03	6.06E−03	8.08E−03	1.35E−04	8.54E−03	3.56E−03	3.71E−04	3.50E−04
f_{16}	Mean	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00	−1.03E+00
	Std	9.18E−11	6.45E−16	5.37E−16	1.08E−04	1.80E−08	3.17E−14	1.96E−09	1.67E−04
f_{17}	Mean	3.97E−01	3.97E−01	3.97E−01	4.00E−01	3.97E−01	3.97E−01	3.97E−01	4.04E−01
	Std	3.63E−11	0	0	2.37E−03	7.21E−07	9.40E−15	5.90E−06	2.62E−02
f_{18}	Mean	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	4.17E+00
	Std	5.5508E−09	1.4659E−15	2.0417E−15	4.2673E−04	4.5341E−05	1.6157E−13	5.6013E−04	3.1285E+00
f_{19}	Mean	−3.86E+00	−3.86E+00	−3.86E+00	−3.85E+00	−3.86E+00	−3.86E+00	−3.85E+00	−3.83E+00
	Std	2.40E−03	2.65E−15	2.44E−15	5.80E−03	3.03E−03	2.86E−10	3.11E−02	2.8777E−02
f_{20}	Mean	−3.28E+00	−3.27E+00	−3.25E+00	−3.05E+00	−3.22E+00	−3.22E+00	−3.21E+00	−2.90E+00
	Std	7.03E−02	5.82E−02	5.99E−02	1.26E−01	7.40E−02	5.92E−02	1.27E−01	1.73E−01
f_{21}	Mean	−1.01E+01	−6.07E+00	−9.40E+00	−4.68E+00	−9.31E+00	−8.31E+00	−8.10E+00	−6.37E+00
	Std	2.43E−08	3.67E+00	2.29E+00	1.54E−01	2.21E+00	3.14E+00	2.53E+00	1.92E+00
f_{22}	Mean	−1.04E+01	−7.95E+00	−1.01E+01	−4.72E+00	−1.04E+01	−8.58E+00	−7.38E+00	−6.29E+00
	Std	1.90E−08	3.53E+00	1.39E+00	1.22E−01	7.34E−04	3.10E+00	3.14E+00	1.94E+00
f_{23}	Mean	−1.05E+01	−6.82E+00	−9.50E+00	−4.77E+00	−1.02E+01	−9.00E+00	−5.95E+00	−6.50E+00
	Std	2.41E−08	3.80E+00	2.67E+00	1.90E−01	1.48E+00	2.89E+00	3.32E+00	2.41E+00
Mean rank		1.6	3.5	2.3	4.8	3.4	2.9	4.2	4.7
Rank		1	5	2	8	4	3	6	7

convergence problem. From Fig. 2, it is clear that VPPSO can avoid premature convergence by balancing exploration and exploitation. Although HGSO has shown fast convergence speed on unimodal functions, it can easily converge to a non-optimal point shortly after the the optimization process starts when it solves multimodal functions. This can be clearly seen in Fig. 2.

4.7 Statistical significance analysis

To statistically validate the effectiveness of VPPSO, two prominent statistical tests are used: Friedman test and Wilcoxon rank-sum test. The Friedman test ranks algorithms for each problem separately. The best algorithm is ranked first while the remaining best algorithms are ranked second, third and so on. From Tables 8-11, it is clear that VPPSO achieves the first rank on unimodal and multimodal functions when tested on low- and high-dimensional cases. To evaluate the overall VPPSO performance, the Friedman mean rank is calculated for all tested functions as shown in Table 17. This table shows that VPPSO achieves the first rank which indicates the superiority of VPPSO.

Wilcoxon rank-sum test is another widely used statistical test to evaluate the significance of novel metaheuristic algorithms or their variants. Considering a 0.05 significance level, the results of a pair-wise comparison between VPPSO and the seven other algorithms are shown in Table 18 for $f_1 - f_{13}$ ($D = 30$) and ($f_{14} - f_{24}$). The results demonstrate that VPPSO is significantly better than other algorithms.

5 Engineering problems

The performance of VPPSO is further evaluated by applying it to solve four well-known engineering optimization problems: welded beam design, speed reducer design, pressure vessel design and tension/compression spring design. Since these four engineering problems have some constraints to be satisfied, particles are divided into valid and invalid ones. A particle that can satisfy all constraints is valid; otherwise it is not. This work follows one of the most common ways to penalize invalid particles in minimization problems where the fitness of each invalid particle is assigned an extremely large value. The parameter settings of all algorithms are exactly the same as in Table 7. The following subsections describe the

Table 10 Statistical results of f_1-f_{13} when $D = 100$

Fun		VPPSO	PSO	PPSO	HGSO	GWO	SSA	WOA	AOA
f_1	Mean	0	1.98E+02	6.94E−01	3.58E−154	1.42E−12	1.45E+03	5.79E−71	2.91E−78
	Std	0	5.53E+01	7.51E−01	1.96E−153	1.15E−12	3.29E+02	3.08E−70	1.47E−77
f_2	Mean	0	1.84E+01	5.61E−01	8.24E−88	4.34E−08	4.88E+01	3.33E−50	4.93E−39
	Std	0	4.49E+01	4.63E−01	4.51E−87	1.68E−08	1.77E+01	1.52E−49	2.67E−38
f_3	Mean	0	4.15E+04	6.64E+00	3.43E−140	7.02E+02	4.95E+04	1.07E+06	5.98E−62
	Std	0	9.42E+03	1.33E+01	1.88E−139	5.98E+02	2.42E+04	2.90E+05	2.29E−61
f_4	Mean	0	2.58E+01	4.06E−02	2.45E−75	6.87E−01	2.80E+01	7.70E+01	6.28E−39
	Std	0	1.95E+00	6.58E−02	1.34E−74	6.12E−01	3.17E+00	2.37E+01	2.27E−38
f_5	Mean	6.26E−01	1.72E+04	1.02E+02	9.86E+01	9.78E+01	1.36E+05	9.80E+01	9.89E+01
	Std	7.47E−01	1.89E+04	4.63E+00	2.95E−01	7.10E−01	6.31E+04	2.80E−01	4.85E−02
f_6	Mean	6.03E−02	2.14E+02	1.16E+01	2.01E+01	1.02E+01	1.41E+03	4.52E+00	2.31E+01
	Std	4.33E−02	8.86E+01	2.56E+00	1.37E+00	9.52E−01	4.85E+02	1.15E+00	3.40E−01
f_7	Mean	3.30E−04	4.29E−01	6.07E−03	2.00E−04	7.75E−03	2.93E+00	4.29E−03	6.30E−04
	Std	4.38E−04	9.48E−02	5.29E−03	1.39E−04	4.06E−03	5.01E−01	4.27E−03	4.51E−04
f_8	Mean	−4.02E+04	−2.02E+04	−2.62E+04	−4.73E+03	−1.63E+04	−2.11E+04	−3.55E+04	−6.66E+03
	Std	2.16E+03	2.06E+03	2.85E+03	7.82E+02	2.48E+03	2.49E+03	5.89E+03	1.00E+03
f_9	Mean	0	2.36E+02	3.31E+00	0	7.16E+00	2.38E+02	0	0
	Std	0	2.43E+01	1.22E+01	0	5.36E+00	4.14E+01	0	0
f_{10}	Mean	7.99E−15	3.83E+00	7.90E−02	8.88E−16	1.33E−07	9.98E+00	4.55E−15	3.01E−15
	Std	0	2.66E−01	5.21E−02	0	5.90E−08	1.15E+00	2.18E−15	1.77E−15
f_{11}	Mean	0	2.79E+00	2.62E−01	0	5.22E−03	1.31E+01	0	0
	Std	0	6.45E−01	2.35E−01	0	1.21E−02	3.87E+00	0	0
f_{12}	Mean	2.61E−03	8.90E+00	1.97E−01	8.27E−01	2.93E−01	3.46E+01	4.14E−02	1.06E+00
	Std	3.72E−03	2.24E+00	5.10E−02	8.47E−02	5.10E−02	1.00E+01	1.55E−02	5.09E−02
f_{13}	Mean	6.25E−02	2.68E+02	1.01E+01	9.93E+00	6.73E+00	6.89E+03	2.46E+00	9.93E+00
	Std	1.27E−01	2.32E+02	7.77E−01	5.32E−02	3.94E−01	1.17E+04	7.65E−01	4.13E−02
Mean rank		1.30	6.23	4.46	3	4.15	7.07	3.30	3.69
Rank		1	7	6	2	5	8	3	4

aforementioned engineering problems and they provide the results of all compared algorithms.

5.1 Welded beam design (WBD)

Welded beam design problem is a well-known engineering benchmark to test the effectiveness of optimization algorithms. The purpose of this design engineering problem is to obtain the best fabrication cost by defining the optimal values of the given variables. The number of variables and constraints in WBD are four and five, respectively. The mathematical representation of WBD is given in Appendix A [90].

The performance of VPPSO on the welded beam design problem is compared with 13 algorithms including CPSO [91], IPSO [92], marine predators algorithm (MPA) [93], GSA, Harris' Hawk optimization [94] and EO. The best fabrication costs achieved by all compared algorithms are recorded in Table 19. In addition, Table 19 shows the best variable values obtained by each algorithm. From Table 19, it is obvious that VPPSO achieves the best cost in comparison with all algorithms.

5.2 Speed reducer design (SRD)

The main objective of this problem is to minimize the weight of speed reducer based on certain constraints associated with diverse components such as gear teeth, bending stress, surface stress, shafts stresses and transverse deflections of the shafts. The SRD problem consists of 7 variables and 11 constraints that must be satisfied. The SRD problem is mathematically written as shown in Appendix B.

Table 20 presents the best variables and the best results achieved by all compared algorithms. Results show that the best weight is achieved by VPPSO.

Table 11 Statistical results of f_1-f_{13} when $D = 500$

Fun		VPPSO	PSO	PPSO	HGSO	GWO	SSA	WOA	AOA
f_1	Mean	0	5.31E+04	5.09E+00	7.32E−164	1.71E−03	9.50E+04	5.64E−68	6.39E−73
	Std	0	6.03E+03	6.55E+00	0	5.68E−04	6.52E+03	2.93E−67	1.87E−72
f_2	Mean	0	1.14E+03	3.19E+00	4.60E−86	1.08E−02	5.35E+02	1.79E−47	7.84E−38
	Std	0	6.08E+01	2.64E+00	2.52E−85	1.56E−03	1.79E+01	5.55E−47	2.02E−37
f_3	Mean	0	1.25E+06	1.03E+04	1.41E−138	3.22E+05	1.36E+06	2.81E+07	2.13E−48
	Std	0	2.51E+05	3.81E+04	7.76E−138	7.66E+04	6.16E+05	8.88E+06	1.17E−47
f_4	Mean	0	5.70E+01	6.02E−02	6.48E−85	6.65E+01	4.08E+01	7.64E+01	7.12E−37
	Std	0	2.81E+00	1.24E−01	3.28E−84	4.56E+00	2.90E+00	2.76E+01	1.82E−36
f_5	Mean	1.13E+01	2.96E+07	5.49E+02	4.98E+02	4.98E+02	3.78E+07	4.96E+02	4.98E+02
	Std	2.26E+01	4.20E+06	5.11E+01	9.77E−02	3.21E−01	5.08E+06	5.09E−01	1.90E−02
f_6	Mean	2.97E+00	5.09E+04	1.19E+02	1.18E+02	9.10E+01	9.67E+04	3.30E+01	1.22E+02
	Std	4.83E+00	4.37E+03	1.09E+01	1.73E+00	1.99E+00	7.05E+03	9.29E+00	5.73E−01
f_7	Mean	3.60E−04	2.20E+02	1.25E−02	2.17E−04	4.72E−02	2.87E+02	2.05E−03	6.45E−04
	Std	4.71E−04	3.81E+01	1.40E−02	2.13E−04	9.85E−03	4.04E+01	1.98E−03	4.53E−04
f_8	Mean	−1.95E+05	−7.08E+04	−6.68E+04	−9.74E+03	−5.74E+04	−5.86E+04	−1.72E+05	−1.48E+04
	Std	1.26E+04	4.69E+03	7.31E+03	2.47E+03	3.81E+03	4.63E+03	2.90E+04	2.05E+03
f_9	Mean	0	3.13E+03	9.52E+00	0	8.22E+01	3.13E+03	3.03E−14	0
	Std	0	1.19E+02	1.14E+01	0	3.08E+01	1.06E+02	1.66E−13	0
f_{10}	Mean	7.99E−15	1.24E+01	2.15E−01	8.88E−16	1.88E−03	1.42E+01	3.84E−15	3.96E−15
	Std	0	6.07E−01	1.42E−01	0	2.64E−04	2.83E−01	1.63E−15	1.22E−15
f_{11}	Mean	0	4.66E+02	6.76E−01	0	1.64E−02	8.28E+02	0	0
	Std	0	4.93E+01	4.10E−01	0	3.36E−02	5.74E+01	0	0
f_{12}	Mean	2.18E−02	8.69E+06	7.26E−01	1.06E+00	7.57E−01	1.52E+06	1.06E−01	1.16E+00
	Std	3.36E−02	2.72E+06	6.05E−02	3.32E−02	7.36E−02	1.01E+06	5.33E−02	1.20E−02
f_{13}	Mean	6.91E−01	5.31E+07	5.38E+01	4.99E+01	5.01E+01	3.41E+07	1.87E+01	4.99E+01
	Std	7.65E−01	1.20E+07	5.46E+00	1.66E−02	1.67E+00	7.64E+06	4.98E+00	5.21E−02
Mean rank		1.30	6.38	4.69	2.84	4.61	6.84	3.23	3.69
Rank		1	7	6	2	5	8	3	4

5.3 Pressure vessel design (PVD)

Pressure vessel design problem is another well-known engineering problem that is used as a benchmark to validate the effectiveness of metaheuristic algorithms. In PVD, the objective is to find the minimal cost of a pressure vessel. PVD is a problem with four variables and four constraints as shown in Appendix C. Table 21 presents the best solutions of all algorithms. It is evident from Table 21 that VPPSO achieves the best result.

5.4 Tension/compression spring design (TSD)

The main objective of this well-known engineering problem is to find the minimum weight of the tension/compression spring while satisfying its design constraints: shear stress, surge frequency and deflection. Three design variables need to be taken into account: wire diameter, mean coil diameter and the number of active coils. The mathematical representation of TSD is given in Appendix D. The performance of VPPSO and the compared algorithms when solving the TSD problem is presented in Table 22. According to the results, VPPSO, PSO, GWO, SSA, WOA, AOA and GSA outperform the other algorithms in terms of finding the minimum weight.

The addition of the third movement option has supported VPPSO to better balance exploration and exploitation. This has been clearly seen in the results provided in this section where VPPSO has shown effective and robust exploration and exploitation abilities in low- and high-dimensional cases. The implementation of a two-swarm strategy has further assisted VPPSO to main diversity and avoid premature convergence. Moreover, the proposed modified velocity equation in VPPSO has played an important role in avoiding undesired rapid movements of particles.

Table 12 Statistical results of the CEC2019 test functions

Fun		VPPSO	PSO	PPSO	HGSO	GWO	SSA	WOA	AOA
f_{24}	Mean	1.00E+00	3.18E+05	5.84E+04	1.00E+00	2.53E+04	2.30E+06	2.15E+07	2.97E+00
	Std	0	4.47E+05	2.94E+05	1.03E−09	6.32E+04	2.69E+06	2.20E+07	1.08E+01
f_{25}	Mean	4.59E+00	3.01E+02	4.61E+01	4.62E+00	4.61E+02	1.51E+03	6.34E+03	5.15E+00
	Std	3.84E−01	9.82E+01	7.90E+01	3.41E−01	2.84E+02	8.91E+02	2.62E+03	1.04E+00
f_{26}	Mean	2.45E+00	3.23E+00	2.76E+00	7.26E+00	2.47E+00	4.73E+00	5.87E+00	5.63E+00
	Std	1.28E+00	1.86E+00	1.48E+00	8.66E−01	1.46E+00	2.13E+00	2.46E+00	8.11E−01
f_{27}	Mean	2.42E+01	3.16E+01	4.78E+01	6.09E+01	1.88E+01	3.47E+01	5.41E+01	5.93E+01
	Std	1.18E+01	1.28E+01	1.99E+01	8.56E+00	1.08E+01	1.94E+01	2.12E+01	1.00E+01
f_{28}	Mean	1.20E+00	1.21E+00	1.63E+00	1.27E+01	2.09E+00	1.22E+00	2.70E+00	4.39E+01
	Std	1.20E−01	1.35E−01	4.11E−01	3.95E+00	9.52E−01	1.45E−01	7.91E−01	1.40E+01
f_{29}	Mean	4.16E+00	4.53E+00	7.19E+00	7.42E+00	2.87E+00	5.02E+00	8.63E+00	8.23E+00
	Std	1.62E+00	1.18E+00	1.77E+00	8.01E−01	9.54E−01	2.15E+00	1.66E+00	1.34E+00
f_{30}	Mean	9.13E+02	1.07E+03	1.17E+03	1.72E+03	9.64E+02	1.04E+03	1.44E+03	1.58E+03
	Std	4.00E+02	3.21E+02	2.79E+02	2.04E+02	3.21E+02	3.80E+02	3.30E+02	2.30E+02
f_{31}	Mean	4.05E+00	4.29E+00	4.50E+00	4.73E+00	4.07E+00	4.34E+00	4.75E+00	4.55E+00
	Std	4.78E−01	3.01E−01	4.43E−01	2.13E−01	5.23E−01	4.30E−01	2.57E−01	2.30E−01
f_{32}	Mean	1.19E+00	1.21E+00	1.38E+00	1.56E+00	1.21E+00	1.40E+00	1.47E+00	2.55E+00
	Std	9.53E−02	1.19E−01	2.17E−01	1.44E−01	7.32E−02	2.01E−01	1.84E−01	6.93E−01
f_{33}	Mean	1.98E+01	2.13E+01	2.10E+01	2.12E+01	2.14E+01	2.10E+01	2.13E+01	2.12E+01
	Std	4.84E+00	1.04E−01	8.31E−02	7.64E−01	8.02E−02	9.84E−02	1.34E−01	3.88E−01
Mean rank		1.2	4	4.2	5.9	3.3	4.5	6.9	6
Rank		1	3	4	6	2	5	8	7

6 Conclusion

A novel PSO variant called Velocity Pausing Particle Swarm Optimization (VPPSO) is proposed in this work. The mean idea of the proposed approach is to provide particles an option that allows them to move with the same velocity in subsequent iterations. The merit of the velocity pausing approach is that it is not limited to PSO variants only but it can also be applied to new or existing meta-heuristic algorithms to improve their performances. VPPSO changes the first term of the standard PSO velocity equation to help avoid the premature convergence of PSO. To enhance diversity, the proposed approach implements a two-swarm strategy where particles in the first swarm update their positions based on the classical PSO mechanism while particles in the second swarm are attracted by the global best position only to update their positions. The performance of VPPSO is validated by testing it on 43 challenging optimization problems: 23 classical benchmark functions, the 10 CEC2019 test functions and the CEC2020 test suite. Moreover, VPPSO is applied to solve four real-world engineering problems. According to the statistical results, VPPSO outperforms recent well-known high-performance optimization algorithms including PPSO, GWO, HGSO and AOA on both low- and high-dimensional problems. This significant VPPSO performance is achieved because the velocity pausing idea can better balance exploration and exploitation. In addition, the two-swarm strategy and the proposed modified velocity equation can help to enhance diversity and better control the movements of particles, respectively. Moreover, VPPSO has shown superior performance when it solves the four real-world constrained engineering problems. These promising results motivate other researchers to apply VPPSO to solve optimization problems in their fields.

7 Future work

Some potential directions that can help to improve the optimization performance of VPPSO and other meta-heuristic algorithms are summarized as follows:

- The velocity pausing concept can be integrated with other metaheuristic algorithms to enhance their performance.
- Further work is need to develop a binary VPPSO version to solve binary optimization problems such as feature selection and the 0–1 knapsack problem.

Table 13 Statistical results of the CEC2020 test functions

Fun		VPPSO	PSO	PPSO	HGSO	GWO	SSA	WOA	AOA
f_{34}	Mean	2.60E+02	4.67E+02	4.67E+02	1.69E+03	2.07E+03	6.35E+02	1.30E+03	3.40E+02
	Std	2.53E+02	1.39E+03	1.39E+03	1.56E+03	2.08E+03	9.25E+02	1.23E+03	4.40E+02
f_{35}	Mean	1.10E+03	1.11E+03	1.10E+03	1.10E+03	1.11E+03	1.10E+03	1.11E+03	1.10E+03
	Std	2.15E−01	3.57E+01	2.41E+01	4.25E−01	3.59E+01	6.81E+00	2.48E+01	3.56E+00
f_{36}	Mean	7.01E+02	7.01E+02	7.01E+02	7.02E+02	7.02E+02	7.01E+02	7.02E+02	7.01E+02
	Std	1.01E+00	7.02E−01	1.02E+00	7.29E−02	4.32E−01	1.03E+00	4.76E−01	4.58E−01
f_{37}	Mean	1.90E+03	1.90E+03	1.90E+03	1.90E+03	1.90E+03	1.90E+03	1.90E+03	1.90E+03
	Std	0	3.60E−03	5.00E−03	0	6.52E−03	5.00E−03	0	0
f_{38}	Mean	2.84E+03	1.82E+03	1.76E+03	2.90E+03	2.96E+03	3.46E+03	3.40E+03	3.09E+03
	Std	1.14E+03	1.21E+02	7.39E+01	1.07E+03	2.40E+03	3.10E+03	2.27E+03	1.52E+03
f_{39}	Mean	1.60E+03	1.62E+03	1.60E+03	1.60E+03	1.60E+03	1.60E+03	1.61E+03	1.60E+03
	Std	6.83E−01	4.50E+01	8.1324	5.95E−01	8.15E−01	7.67E+00	3.68E+01	1.20E+00
f_{40}	Mean	2.79E+03	2.20E+03	2.15E+03	3.42E+03	3.83E+03	3.59E+03	5.57E+03	3.55E+03
	Std	7.05E+02	1.17E+02	1.43E+02	7.90E+02	1.31E+03	2.15E+03	5.19E+03	3.75E+03
f_{41}	Mean	2.23E+03	2.26E+03	2.25E+03	2.28E+03	2.27E+03	2.20E+03	2.27E+03	2.30E+03
	Std	3.90E+01	4.51E+01	4.39E+01	3.79E+01	4.45E+01	6.21E+00	4.45E+01	3.04E+01
f_{42}	Mean	2.55E+03	2.58E+03	2.61E+03	2.51E+03	2.59E+03	2.56E+03	2.62E+03	2.55E+03
	Std	9.16E+01	1.21E+02	1.14E+02	4.18E+00	1.02E+02	1.07E+02	1.07E+02	4.77E+01
f_{43}	Mean	2.83E+03	2.84E+03	2.84E+03	2.85E+03	2.84E+03	2.84E+03	2.84E+03	2.86E+03
	Std	6.39E+01	1.06E−02	3.03E−02	2.61E+00	3.28E−02	1.44E+01	1.42E+01	1.38E+01
Mean rank		1.6	4	3.3	4.5	5.1	3.8	6	4.4
Rank		1	4	2	6	7	3	8	5

- Another interesting future work is the development of a multi-objective VPPSO algorithm.
- VPPSO can be hybridized with other recent algorithms such as EO, HGSO and AOA to further improve its performance.
- In terms of applications, VPPSO can be applied to solve diverse real-world optimization problems such as maintenance scheduling [100], data clustering [101], lot-sizing optimization [102, 103] and multilevel thresholding image segmentation [14, 104].
- One potential direction is to combine VPPSO with well-known approaches such as Levy flight and chaotic maps to develop an enhanced version of VPPSO.
- VPPSO can be applied to optimize real-world engineering problems such as three-bar truss design and multiple disc clutch brake.

Table 14 Statistical results for different values of α in the first case where $N_1 = 20$ and $N_2 = 10$

Fun		$\alpha = 1$	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$	$\alpha = 0.4$	$\alpha = 0.3$	$\alpha = 0.2$	$\alpha = 0.1$
f_1	Mean	1.0740E−13	4.0145E−14	1.5206E−13	3.4060E−13	0	0	0	0	0	0
	Std	2.0235E−13	1.3709E−13	4.2009E−13	1.5731E−12	0	0	0	0	0	0
f_2	Mean	4.2250E−09	6.1076E−09	3.0117E−09	2.1381E−09	0	0	0	0	0	0
	Std	4.9789E−09	7.7185E−09	3.7862E−09	3.6237E−09	0	0	0	0	0	0
f_3	Mean	2.8633E+00	8.1757E+00	7.8212E+00	5.3997E+00	2.6922E+00	1.7731E−08	0	0	0	0
	Std	5.2307E+00	1.9636E+01	3.1985E+01	2.0412E+01	1.4200E+01	9.7118E−08	0	0	0	0
f_4	Mean	5.4272E+00	4.5572E+00	5.8247E+00	3.4271E+00	1.1082E−03	0	0	0	0	0
	Std	3.9745E+00	4.0583E+00	4.9080E+00	4.1303E+00	6.0698E−03	0	0	0	0	0
f_5	Mean	6.0447E−03	8.1068E−03	8.8890E−03	7.5756E−01	5.8569E−02	1.7444E−02	7.8419E−03	7.1545E−03	5.8675E−03	4.3621E−02
	Std	5.5676E−03	7.4615E−03	8.5286E−03	4.0798E+00	2.1692E−01	3.1156E−02	8.3687E−03	1.0755E−02	9.8910E−03	4.8610E−02
f_6	Mean	2.8275E−07	2.8580E−07	3.2120E−07	3.3974E−07	2.7429E−07	3.0593E−07	3.1530E−07	2.6991E−07	2.7181E−07	2.9952E−07
	Std	1.3371E−07	1.6852E−07	1.5898E−07	2.1855E−07	1.0753E−07	1.4040E−07	3.4039E−07	9.2880E−08	1.1894E−07	1.1076E−07
f_7	Mean	3.8890E−03	4.5037E−03	2.1717E−03	3.0567E−03	2.6610E−03	1.8660E−03	8.4429E−04	6.2221E−04	3.8730E−04	3.7386E−04
	Std	4.1993E−03	4.8191E−03	1.3465E−03	2.5313E−03	3.6356E−03	1.9260E−03	7.6180E−04	7.1045E−04	5.5741E−04	6.0076E−04
f_8	Mean	−6.2012E+03	−6.2384E+03	−7.5729E+03	−8.7569E+03	−1.0521E+04	-1.1114E+04	-1.1989E+04	-1.2274E+04	-1.2466E+04	-1.2447E+04
	Std	7.6169E+02	1.0194E+03	1.7018E+03	2.1649E+03	1.8961E+03	1.4240E+03	6.7207E+02	4.6183E+02	2.1362E+02	1.7434E+02
f_9	Mean	8.8568E+00	2.2221E+00	9.9499E−02	1.5256E+00	0	0	0	0	0	0
	Std	2.5930E+01	7.9540E+00	4.0056E−01	5.8585E+00	0	0	0	0	0	0
f_{10}	Mean	5.2590E−08	4.2238E−08	1.9264E−08	5.6178E−09	7.9936E−15	7.9936E−15	7.9936E−15	7.8752E−15	7.8752E−15	7.9936E−15
	Std	7.2909E−08	3.1099E−08	2.2403E−08	1.5988E−08	0	0	0	6.4863E−16	6.4863E−16	0
f_{11}	Mean	2.2118E−02	5.5788E−03	7.1351E−03	1.0381E−02	1.3922E−03	0	0	0	0	0
	Std	2.9546E−02	2.1881E−02	1.7045E−02	2.4659E−02	7.6252E−03	0	0	0	0	0
f_{12}	Mean	2.4209E+00	2.7007E+00	1.2809E+00	4.5324E−01	1.0703E−02	3.4933E−03	6.5395E−06	6.9466E−03	4.8446E−06	4.0187E−05
	Std	4.0173E+00	5.3992E+00	2.6552E+00	1.5168E+00	5.8300E−02	1.9080E−02	1.3542E−05	2.6417E−02	3.5470E−06	1.2586E−04
f_{13}	Mean	2.5256E−02	4.4718E−03	2.2734E−02	1.4218E−02	2.9949E−02	1.3036E−02	1.1963E−03	1.3615E−03	1.8788E−03	3.3066E−03
	Std	4.1779E−02	5.5847E−03	3.9269E−02	2.2138E−02	5.6045E−02	2.7188E−02	3.1580E−03	3.5435E−03	4.1811E−03	7.1089E−03
f_{14}	Mean	7.1813E+00	2.8454E+00	1.8571E+00	1.4615E+00	1.1970E+00	1.0315E+00	1.2981E+00	1.0982E+00	1.6299E+00	2.2853E+00
	Std	4.8467E+00	1.5920E+00	1.1534E+00	6.7646E−01	4.0432E−01	1.8142E−01	4.6214E−01	3.0306E−01	7.1473E−01	1.4669E+00
f_{15}	Mean	5.1770E−03	1.9961E−03	3.2482E−03	1.9941E−03	1.8435E−03	1.1929E−03	2.5541E−03	1.7698E−03	2.5135E−03	5.1743E−03
	Std	8.5227E−03	5.0027E−03	6.8369E−03	5.0047E−03	5.0376E−03	3.6333E−03	5.9814E−03	5.0640E−03	6.0731E−03	8.5381E−03

Table 14 (continued)

Fun		$\alpha = 1$	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$	$\alpha = 0.4$	$\alpha = 0.3$	$\alpha = 0.2$	$\alpha = 0.1$
f_{16}	Mean	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00
	Std	1.6651E−10	2.3807E−10	1.7095E−10	1.0108E−10	2.0195E−10	1.8283E−10	1.5641E−10	2.1082E−10	1.8790E−10	1.7536E−10
f_{17}	Mean	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01
	Std	5.7342E−11	5.9843E−11	6.0181E−11	6.1676E−11	6.6495E−11	5.6099E−11	7.1232E−11	5.6696E−11	6.4089E−11	7.9035E−11
f_{18}	Mean	3.0000E+00	5.7000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00
	Std	6.9077E−09	1.4789E+01	8.2635E−09	1.1135E−08	7.1408E−09	1.1996E−08	8.4349E−09	1.0176E−08	1.1671E−08	8.8019E−09
f_{19}	Mean	−3.8617E+00	−3.8615E+00	−3.8625E+00	−3.8615E+00	−3.8623E+00	−3.8625E+00	−3.8620E+00	−3.8620E+00	−3.8604E+00	−3.8612E+00
	Std	2.7250E−03	2.9875E−03	1.4390E−03	2.9875E−03	1.9996E−03	1.4389E−03	2.4048E−03	2.4049E−03	3.6735E−03	3.2044E−03
f_{20}	Mean	−3.2471E+00	−3.2437E+00	−3.2628E+00	−3.2613E+00	−3.2707E+00	−3.2380E+00	−3.2463E+00	−3.2758E+00	−3.2671E+00	−3.2311E+00
	Std	9.3462E−02	7.7281E−02	7.6444E−02	7.1946E−02	7.0124E−02	7.3184E−02	8.0421E−02	6.7850E−02	8.8189E−02	9.8458E−02
f_{21}	Mean	−5.7782E+00	−6.4518E+00	−8.7273E+00	-1.0153E+01	-1.0153E+01	-1.0153E+01	-1.0153E+01	-1.0153E+01	-1.0153E+01	-1.0153E+01
	Std	2.0850E+00	2.5406E+00	2.6966E+00	2.7764E−08	2.2884E−08	3.8921E−08	2.6335E−08	2.4826E−08	4.1770E−08	2.6843E−08
f_{22}	Mean	−6.4565E+00	−7.0809E+00	−9.0935E+00	-1.0148E+01	-1.0403E+01	-1.0403E+01	-1.0403E+01	-1.0403E+01	-1.0403E+01	-1.0403E+01
	Std	2.4582E+00	2.8201E+00	2.4480E+00	1.3943E+00	2.4125E−08	2.8206E−08	2.9215E−08	2.4859E−08	2.7235E−08	2.8861E−08
f_{23}	Mean	−7.4221E+00	−8.1597E+00	−9.9214E+00	−9.7381E+00	-1.0536E+01	-1.0536E+01	-1.0536E+01	-1.0536E+01	-1.0536E+01	-1.0536E+01
	Std	3.2821E+00	3.2742E+00	1.9107E+00	2.4364E+00	2.6337E−08	3.4521E−08	2.6350E−08	2.0585E−08	2.9648E−08	2.3076E−08
Mean rank		6.39	6.78	5.78	5.26 3.95	3.91	3.39	2.39	3.65	4.08	
Rank		9	10	8	7	5	4	2	1	3	6

Table 15 Statistical results for different values of α in the second case where $N_1 = 15$ and $N_2 = 15$

Fun		$\alpha = 1$	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$	$\alpha = 0.4$	$\alpha = 0.3$	$\alpha = 0.2$	$\alpha = 0.1$
f_1	Mean	1.9374E−15	2.2719E−15	1.7405E−15	2.4947E−15	9.5265E−15	0	0	0	0	0
	Std	3.5265E−15	3.9372E−15	4.2407E−15	6.7856E−15	5.0728E−14	0	0	0	0	0
f_2	Mean	1.0732E−09	1.2999E−09	9.2905E−10	1.7926E−10	1.5220E−11	0	0	0	0	0
	Std	8.0486E−10	1.0807E−09	7.7864E−10	3.3271E−10	8.3364E−11	0	0	0	0	0
f_3	Mean	5.2597E−02	1.1285E+00	3.7903E−02	3.3745E+00	3.9994E−04	0	0	0	0	0
	Std	1.5606E−01	6.0683E+00	1.7911E−01	1.5411E+01	2.1266E−03	0	0	0	0	0
f_4	Mean	5.1899E+00	2.5321E+00	2.9192E+00	2.0616E+00	2.8039E−05	0	0	0	0	0
	Std	4.9353E+00	3.9449E+00	4.5093E+00	3.7944E+00	1.5311E−04	0	0	0	0	0
f_5	Mean	9.2687E−01	1.7951E+00	1.9827E−03	1.2686E+00	9.1318E−01	5.1129E−03	3.7902E−03	1.2988E−03	2.0486E−03	1.1921E−02
	Std	5.0061E+00	6.8197E+00	2.0695E−03	5.1147E+00	4.9749E+00	8.2990E−03	6.4773E−03	1.5295E−03	2.5883E−03	2.4939E−02
f_6	Mean	1.1723E−07	1.3477E−07	1.2284E−07	1.2388E−07	1.2343E−07	1.2960E−07	1.2401E−07	1.2068E−07	1.3140E−07	1.3739E−07
	Std	2.8505E−08	4.0992E−08	2.9665E−08	3.2934E−08	3.2684E−08	3.9205E−08	3.2376E−08	3.6333E−08	3.6238E−08	2.9056E−08
f_7	Mean	2.7852E−03	1.9928E−03	2.5907E−03	1.5174E−03	1.8988E−03	1.6807E−03	6.5992E−04	6.1031E−04	8.5365E−04	7.8121E−04
	Std	2.8553E−03	2.2521E−03	2.4650E−03	1.4783E−03	1.8640E−03	2.0296E−03	6.5307E−04	5.9569E−04	9.1302E−04	1.0540E−03
f_8	Mean	−5.9200E+03	−6.4848E+03	−8.0746E+03	−8.9093E+03	-1.0410E+04	-1.1145E+04	-1.1803E+04	-1.2251E+04	-1.2302E+04	-1.2239E+04
	Std	9.9769E+02	7.2290E+02	2.2254E+03	2.5226E+03	1.5581E+03	1.5315E+03	9.8645E+02	5.0009E+02	4.8339E+02	5.0516E+02
f_9	Mean	2.9185E+00	1.9899E+00	6.6332E−02	3.3165E−02	0	0	0	0	0	0
	Std	1.5427E+01	8.7327E+00	3.6331E−01	1.8165E−01	0	0	0	0	0	0
f_{10}	Mean	8.3437E−09	9.0255E−09	6.4641E−09	3.9685E−09	7.9936E−15	7.9936E−15	7.9936E−15	7.9936E−15	7.9936E−15	7.9936E−15
	Std	6.7952E−09	9.0201E−09	7.5267E−09	1.0572E−08	0	0	0	0	0	0
f_{11}	Mean	6.0531E−03	8.3580E−03	5.1429E−03	7.2723E−03	5.1437E−03	0	0	0	0	0
	Std	1.5223E−02	1.8699E−02	1.4754E−02	2.2518E−02	1.7042E−02	0	0	0	0	0
f_{12}	Mean	1.6093E+00	4.3698E−01	3.9376E−01	3.2573E−01	1.4536E−01	1.4070E−02	1.0562E−02	1.8362E−07	1.7063E−07	1.5301E−07
	Std	3.2380E+00	1.6488E+00	1.4982E+00	9.7608E−01	7.2814E−01	3.6212E−02	3.2230E−02	3.5270E−07	4.9592E−07	2.0952E−07
f_{13}	Mean	7.2899E−03	5.4330E−03	7.5984E−03	6.7859E−03	1.0202E−02	2.7890E−02	5.0768E−03	1.8326E−03	1.1004E−03	3.7845E−03
	Std	1.1930E−02	6.7672E−03	9.8457E−03	1.2016E−02	1.8422E−02	6.2506E−02	8.4177E−03	4.1649E−03	3.3529E−03	5.2238E−03
f_{14}	Mean	5.9863E+00	2.2207E+00	2.3476E+00	1.7919E+00	1.3117E+00	1.0973E+00	1.1843E+00	1.1366E+00	1.8583E+00	2.5892E+00
	Std	4.5195E+00	1.1813E+00	2.0013E+00	9.1702E−01	6.4395E−01	3.9953E−01	4.3045E−01	3.2975E−01	1.0295E+00	1.6811E+00
f_{15}	Mean	3.2236E−03	3.2961E−03	7.0227E−03	3.8371E−03	1.2021E−03	3.2132E−03	3.6950E−03	1.1585E−03	1.0530E−03	3.2242E−03
	Std	6.8486E−03	6.8292E−03	1.2633E−02	7.5225E−03	3.6339E−03	6.8488E−03	6.9429E−03	3.6371E−03	1.6866E−03	6.8611E−03
f_{16}	Mean	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00

Table 15 (continued)

Fun		$\alpha = 1$	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$	$\alpha = 0.4$	$\alpha = 0.3$	$\alpha = 0.2$	$\alpha = 0.1$
	Std	9.7709E−11	8.8795E−11	7.9677E−11	1.0590E−10	9.6686E−11	1.3403E−10	1.0114E−10	9.1808E−11	1.0290E−10	9.2493E−11
f_{17}	Mean	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01
	Std	3.4048E−11	6.7741E−11	2.2481E−11	4.8564E−11	5.2777E−11	5.7202E−11	2.9531E−11	3.6394E−11	3.8075E−11	3.6193E−11
f_{18}	Mean	3.0000E+00	5.7000E+00	3.0000E+00	3.0000E+00	5.7000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00
	Std	4.6211E−09	1.4789E+01	5.7280E−09	7.6949E−09	1.4789E+01	5.0998E−09	4.2154E−09	5.5508E−09	5.8122E−09	6.7770E−09
f_{19}	Mean	−3.8620E+00	−3.8623E+00	−3.8620E+00	−3.8620E+00	−3.8625E+00	−3.8620E+00	−3.8623E+00	−3.8620E+00	−3.8607E+00	−3.8609E+00
	Std	2.4049E−03	1.9996E−03	2.4049E−03	2.4049E−03	1.4390E−03	2.4049E−03	1.9996E−03	2.4049E−03	3.5449E−03	3.3905E−03
f_{20}	Mean	−3.2669E+00	−3.2548E+00	−3.2737E+00	−3.2668E+00	−3.2684E+00	−3.2684E+00	−3.2781E+00	−3.2823E+00	−3.2481E+00	−3.2583E+00
	Std	8.3295E−02	7.0039E−02	6.7843E−02	7.0335E−02	7.3399E−02	7.8640E−02	6.4059E−02	7.0374E−02	1.0486E−01	9.2986E−02
f_{21}	Mean	−6.8747E+00	−6.9571E+00	−8.2149E+00	-1.0153E+01	-1.0153E+01	-1.0153E+01	-1.0153E+01	-1.0153E+01	-1.0153E+01	-1.0153E+01
	Std	2.8176E+00	2.7238E+00	2.8515E+00	2.0222E−08	2.5285E−08	2.0311E−08	2.3774E−08	2.4398E−08	1.7969E−08	1.9766E−08
f_{22}	Mean	−7.3867E+00	−8.1048E+00	−8.7860E+00	-1.0403E+01	-1.0403E+01	-1.0403E+01	-1.0403E+01	-1.0403E+01	-1.0403E+01	-1.0403E+01
	Std	2.9435E+00	2.9444E+00	2.7664E+00	1.7795E−08	2.4488E−08	2.8742E−08	2.5889E−08	1.9087E−08	2.3641E−08	2.0682E−08
f_{23}	Mean	−7.6809E+00	−7.0875E+00	−8.9392E+00	-1.0536E+01	-1.0536E+01	-1.0536E+01	-1.0536E+01	-1.0536E+01	-1.0536E+01	-1.0536E+01
	Std	3.2152E+00	3.1945E+00	2.8781E+00	2.0320E−08	2.1296E−08	2.6814E−08	2.9676E−08	2.4145E−08	2.3093E−08	1.9142E−08
Mean rank		6.21	7.13	5.21	5.34	4.13	4.43	2.95	2.26	3.60	4.13
Rank		8	9	6	7	4	5	2	1	3	4

Table 16 Statistical results for different values of α in the third case where $N_1 = 10$ and $N_2 = 20$

Fun		$\alpha = 1$	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$	$\alpha = 0.4$	$\alpha = 0.3$	$\alpha = 0.2$	$\alpha = 0.1$
f_1	Mean	2.5121E−16	8.2320E−16	3.6067E−16	1.4124E−15	8.1269E−18	1.4954E−17	0	0	0	0
	Std	4.1866E−16	2.1538E−15	7.0600E−16	7.2895E−15	2.0641E−17	8.0068E−17	0	0	0	0
f_2	Mean	6.3697E−10	3.9895E−10	3.7706E−10	1.2174E−10	0	0	0	0	0	0
	Std	9.4659E−10	3.8340E−10	5.2586E−10	2.2511E−10	0	0	0	0	0	0
f_3	Mean	4.1409E−04	6.6368E−06	5.3330E−04	2.6498E−04	7.3863E−04	1.8334E−06	0	0	0	0
	Std	1.9116E−03	1.5364E−05	2.9180E−03	1.3279E−03	4.0456E−03	1.0042E−05	0	0	0	0
f_4	Mean	3.1725E+00	1.2571E+00	7.6465E−01	3.1121E−01	8.7220E−02	0	0	0	0	0
	Std	5.9970E+00	2.8446E+00	1.9112E+00	9.9525E−01	3.9918E−01	0	0	0	0	0
f_5	Mean	1.8251E−03	4.4972E+00	1.8723E−03	3.6000E+00	2.5705E−03	1.2947E−01	1.6114E−03	1.8823E−03	1.7614E−03	8.6854E−01
	Std	3.6768E−03	1.0226E+01	3.9590E−03	9.3330E+00	3.3448E−03	6.9804E−01	3.2011E−03	2.4514E−03	1.9719E−03	4.7400E+00
f_6	Mean	9.1490E−08	9.3538E−08	9.3956E−08	8.8880E−08	8.1242E−08	8.7159E−08	9.2792E−08	8.7128E−08	9.9317E−08	9.4414E−08
	Std	2.2931E−08	2.1221E−08	2.4633E−08	2.5100E−08	1.8871E−08	2.1113E−08	1.9279E−08	1.9848E−08	2.9986E−08	2.0284E−08
f_7	Mean	1.5187E−03	1.3831E−03	1.4169E−03	2.0232E−03	1.1657E−03	1.2284E−03	1.1771E−03	7.6457E−04	4.2182E−04	8.4742E−04
	Std	1.5037E−03	1.1988E−03	1.6555E−03	2.0250E−03	9.0124E−04	9.9643E−04	1.1526E−03	6.6793E−04	3.6156E−04	1.0318E−03
f_8	Mean	−5.8041E+03	−6.2690E+03	−7.1279E+03	−8.7990E+03	−9.9851E+03	-1.0775E+04	-1.1643E+04	-1.2036E+04	-1.2433E+04	-1.1927E+04
	Std	8.5762E+02	8.7558E+02	2.0044E+03	2.4814E+03	1.9672E+03	1.5253E+03	9.8901E+02	9.4042E+02	2.2938E+02	8.5242E+02
f_9	Mean	9.1707E−12	1.0232E−13	3.0070E−12	0	0	0	0	0	0	0
	Std	3.8099E−11	2.8221E−13	1.6438E−11	0	0	0	0	0	0	0
f_{10}	Mean	4.3331E−09	4.1528E−09	1.9643E−09	8.9440E−10	8.5578E−10	7.8752E−15	7.9936E−15	7.9936E−15	7.9936E−15	7.9936E−15
	Std	5.7234E−09	4.1141E−09	2.0667E−09	1.5607E−09	2.4481E−09	6.4863E−16	0	0	0	0
f_{11}	Mean	3.2882E−04	6.9425E−03	6.4530E−03	1.6407E−03	6.5732E−04	3.3307E−16	0	0	0	0
	Std	1.8010E−03	2.7126E−02	1.9971E−02	5.6446E−03	3.6003E−03	1.8243E−15	0	0	0	0
f_{12}	Mean	4.8557E−01	2.1525E−02	3.2943E−01	1.4504E−02	1.0587E−02	1.5338E−08	5.0655E−08	9.3040E−09	1.6797E−08	1.1324E−08
	Std	1.5954E+00	5.3316E−02	1.0514E+00	4.8088E−02	3.1587E−02	2.0732E−08	1.7885E−07	5.8304E−09	1.7678E−08	1.0854E−08
f_{13}	Mean	3.6434E−03	6.5610E−03	3.9656E−03	1.1213E−02	1.5636E−02	1.7505E−02	1.4213E−02	3.6311E−03	7.9193E−03	3.6310E−03
	Std	5.9248E−03	1.1715E−02	6.5849E−03	2.5858E−02	2.8634E−02	4.7252E−02	2.0503E−02	5.9086E−03	1.7396E−02	5.9084E−03
f_{14}	Mean	7.0522E+00	2.7382E+00	2.4468E+00	2.0228E+00	1.7277E+00	1.1969E+00	1.3518E+00	1.4662E+00	2.0225E+00	2.4761E+00
	Std	4.8122E+00	2.5912E+00	1.9889E+00	1.0879E+00	8.9859E−01	4.0438E−01	6.3221E−01	4.9826E−01	1.1465E+00	1.5057E+00
f_{15}	Mean	9.6520E−03	6.3590E−03	3.7094E−03	4.4880E−03	3.1544E−03	2.7235E−03	2.0332E−03	4.6376E−03	2.4180E−03	6.4126E−03
	Std	1.3090E−02	9.3251E−03	1.1177E−02	8.0799E−03	6.8689E−03	6.1515E−03	5.0793E−03	8.0076E−03	6.0875E−03	1.3015E−02

Table 16 (continued)

Fun		$\alpha = 1$	$\alpha = 0.9$	$\alpha = 0.8$	$\alpha = 0.7$	$\alpha = 0.6$	$\alpha = 0.5$	$\alpha = 0.4$	$\alpha = 0.3$	$\alpha = 0.2$	$\alpha = 0.1$
f_{16}	Mean	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0316E+00	-1.0306E+00	-1.0316E+00
	Std	1.0299E−10	8.8139E−11	8.7979E−11	4.9732E−11	7.7253E−11	1.0038E−10	8.3121E−11	5.4914E−11	5.7745E−03	5.7405E−11
f_{17}	Mean	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01	3.9789E−01
	Std	2.7556E−11	3.0695E−11	2.3529E−11	4.1375E−11	4.2908E−11	3.2757E−11	2.7944E−11	3.7616E−11	4.6174E−11	3.1971E−11
f_{18}	Mean	5.7000E+00	1.3800E+01	1.3800E+01	5.7000E+00	8.4000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00	3.0000E+00
	Std	1.4789E+01	2.8005E+01	2.8005E+01	1.4789E+01	2.0550E+01	3.6817E−09	4.6391E−09	2.5707E−09	4.7419E−09	4.4370E−09
f_{19}	Mean	−3.8620E+00	−3.8623E+00	−3.8620E+00	−3.8617E+00	−3.8623E+00	−3.8620E+00	−3.8612E+00	−3.8612E+00	−3.8612E+00	−3.8617E+00
	Std	2.4049E−03	1.9996E−03	2.4049E−03	2.7250E−03	1.9996E−03	2.4049E−03	3.2065E−03	3.2065E−03	3.2065E−03	2.7250E−03
f_{20}	Mean	−3.2478E+00	−3.2674E+00	−3.2770E+00	−3.2852E+00	−3.2613E+00	−3.2931E+00	−3.2646E+00	−3.2665E+00	−3.2577E+00	−3.2355E+00
	Std	8.0733E−02	7.6878E−02	7.3926E−02	6.9561E−02	8.3248E−02	6.7791E−02	8.6651E−02	8.2808E−02	8.9386E−02	1.0067E−01
f_{21}	Mean	−7.1255E+00	−6.5342E+00	−8.0499E+00	−9.9848E+00	-1.0153E+01	-1.0153E+01	-1.0153E+01	-1.0153E+01	-1.0153E+01	-1.0153E+01
	Std	2.7610E+00	2.4510E+00	2.6566E+00	9.2244E−01	1.5778E−08	1.6532E−08	2.1568E−08	2.3057E−08	1.8395E−08	1.9354E−08
f_{22}	Mean	−8.6951E+00	−7.9600E+00	−9.4171E+00	-1.0227E+01	-1.0148E+01	-1.0403E+01	-1.0403E+01	-1.0403E+01	-1.0403E+01	-1.0403E+01
	Std	2.6919E+00	2.8982E+00	2.3250E+00	9.6292E−01	1.3943E+00	1.9479E−08	1.6380E−08	2.2369E−08	2.2340E−08	1.9027E−08
f_{23}	Mean	−7.9192E+00	−8.4792E+00	−8.9237E+00	-1.0008E+01	-1.0536E+01	-1.0536E+01	-1.0536E+01	-1.0536E+01	-1.0536E+01	-1.0536E+01
	Std	3.1256E+00	2.7927E+00	2.7911E+00	2.0098E+00	2.3032E−08	1.9394E−08	1.7406E−08	1.7871E−08	2.0041E−08	1.8913E−08
Mean rank		6.65	6.43	5.82	5.56	4.56	3.65	3.26	2.65	4.04	3.78
Rank		10	9	8	7	6	3	2	1	5	4

Table 17 Friedman test result

	VPPSO	PSO	PPSO	HGSO	GWO	SSA	WOA	AOA
Mean Rank	1.3768	5	4.1159	3.8986	4.1739	5.4493	4.4928	4.4348
Rank	1	7	3	2	4	8	6	5

Appendix A: Welded beam design problem

$$\begin{aligned}
\min_x \quad & f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2) \\
\text{s.t.} \quad & g_1(x) = \tau(x) - \tau_{max} \le 0 \\
& g_2(x) = \sigma(x) - \sigma_{max} \le 0 \\
& g_3(x) = x_1 - x_4 \le 0 \\
& g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \le 0 \\
& g_5(x) = 0.125 - x_1 \le 0 \\
& g_6(x) = \delta(x) - \delta_{max} \le 0 \\
& g_7(x) = P - P_c(x) \le 0 \\
\text{range} \quad & 0.1 \le x_i \le 2 \quad i = 1,4 \\
& 0.1 \le x_i \le 10 \quad i = 2,3 \\
\text{where} \quad & \tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \\
& \tau' = \frac{P}{\sqrt{2}x_1x_2}, \quad \tau'' = \frac{MR}{J} \\
& M = P(L + \frac{x_2}{2}) \\
& R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \\
& J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \\
& \sigma(x) = \frac{6PL}{x_4x_3^2}, \quad \delta(x) = \frac{4PL^3}{Ex_3^3x_4} \\
& P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right), \\
& \tau_{max} = 13600psi, \quad \sigma_{max} = 30000psi, \\
& \delta_{max} = 0.25in, \quad P = 6000lb \\
& E = 30 \times 10^6psi, \quad L = 14in, \quad G = 12 \times 10^6psi
\end{aligned}$$

Appendix B: Speed reducer design problem

$$\begin{aligned}
\min_x \quad & f(x) = 0.7854x_1x_2^2\left(3.3333x_3^2 + 14.9334x_3 - 43.0934\right) \\
& - 1.508x_1\left(x_6^2 + x_7^2\right) + 7.4777\left(x_6^3 + x_7^3\right) \\
& + 0.7854\left(x_4x_6^2 + x_5x_7^2\right) \\
\text{s.t.} \quad & g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \le 0 \\
& g_2(x) = \frac{397.5}{x_1x_2^2x_3} - 1 \le 0 \\
& g_3(x) = \frac{1.93x_4^3}{x_2x_6^4x_3} - 1 \le 0 \\
& g_4(x) = \frac{1.93x_5^3}{x_2x_7^4x_3} - 1 \le 0 \\
& g_5(x) = \frac{\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6}}{110x_6^3} - 1 \le 0 \\
& g_6(x) = \frac{\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6}}{85x_7^3} - 1 \le 0 \\
& g_7(x) = \frac{x_2x_3}{40} - 1 \le 0 \\
& g_8(x) = \frac{5x_2}{x_1} - 1 \le 0 \\
& g_9(x) = \frac{x_1}{12x_2} - 1 \le 0 \\
& g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0 \\
& g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0 \\
\text{range} \quad & 2.6 \le x_1 \le 3.6, \quad 0.7 \le x_2 \le 0.8, \quad 17 \le x_3 \le 28, \\
& 7.3 \le x_4 \le 8.3, \quad 7.3 \le x_5 \le 8.3, \quad 2.9 \le x_6 \le 3.9,
\end{aligned}$$

$5.0 \le x_7 \le 5.5$

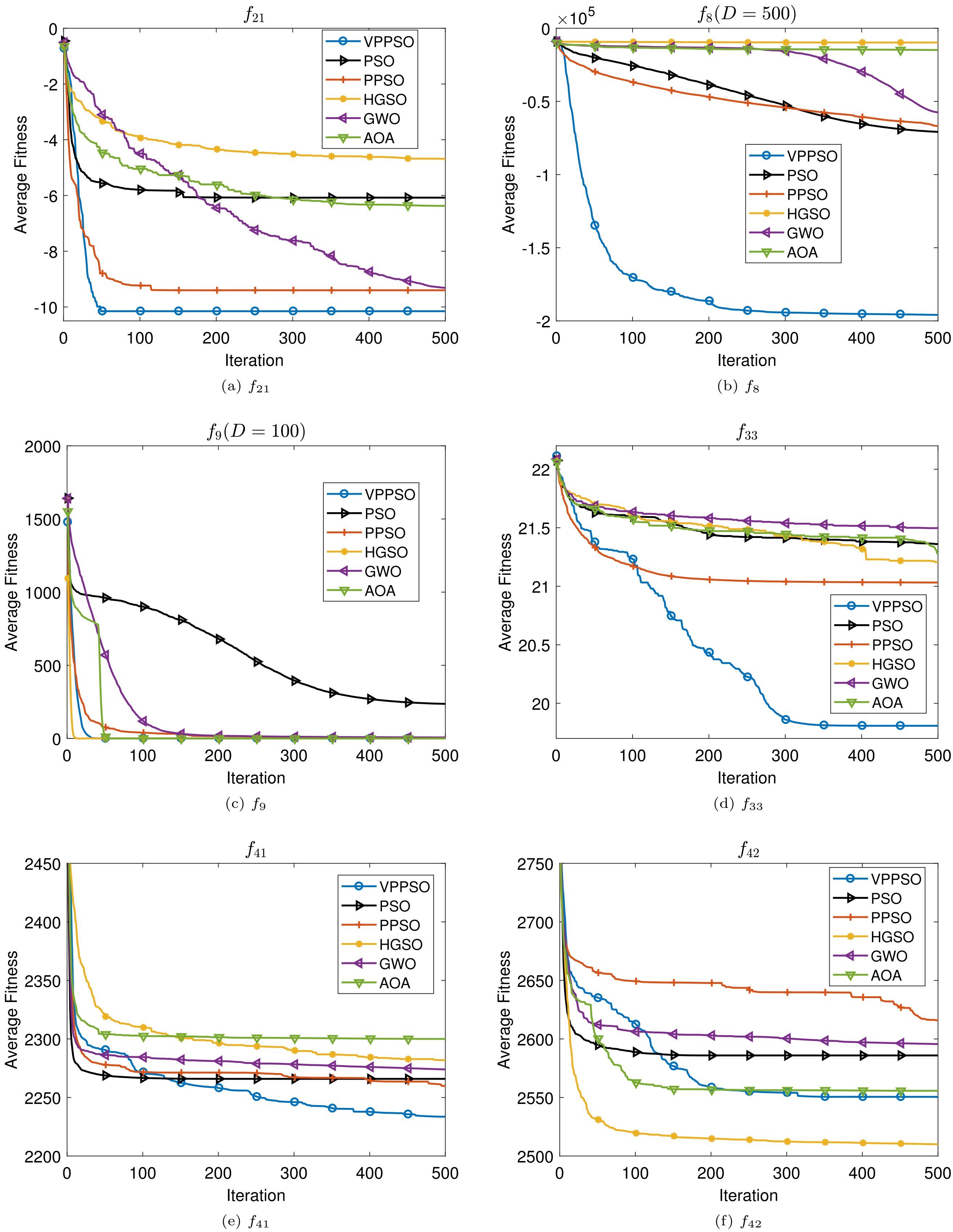

Fig. 2 Convergence curves for some of the benchmarking functions

Table 18 The p values obtained by the Wilcoxon rank-sum test at 0.05 significance level for VPPSO against the seven compared algorithms for $f_1 - f_{13}$ when $D = 30$ and $f_{14} - -f_{23}$

Fun	PSO	PPSO	HGSO	GWO	SSA	WOA	AOA
f_1	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12
f_2	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12
f_3	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12
f_4	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12	1.2118E−12
f_5	3.0199E−11	3.0199E−11	3.0199E−11	3.0199E−11	3.0199E−11	3.0199E−11	3.0199E−11
f_6	3.0199E−11	3.0199E−11	3.0199E−11	3.0199E−11	4.6756E−02	3.0199E−11	3.0199E−11
f_7	3.0199E−11	5.8587E−06	2.2658E−03	6.0104E−08	3.0199E−11	1.7290E−06	**1.1199E−01**
f_8	3.0199E−11	4.1997E−10	8.4848E−09	3.0199E−11	3.0199E−11	6.0459E−07	3.0199E−11
f_9	1.2118E−12	1.2118E−12	**NaN**	1.1970E−12	1.2118E−12	**NaN**	**NaN**
f_{10}	1.2118E−12	1.2118E−12	2.7085E−14	1.1795E−12	1.2118E−12	1.0793E−09	4.6350E−13
f_{11}	1.2118E−12	1.2118E−12	**NaN**	1.1035E−02	1.2118E−12	8.1523E−02	**3.3371E−01**
f_{12}	4.9752E−11	3.0199E−11	3.0199E−11	3.0199E−11	3.0199E−11	3.0199E−11	3.0199E−11
f_{13}	3.3520E−08	3.0199E−11	3.0199E−11	3.0199E−11	4.9752E−11	3.0199E−11	3.0199E−11
f_{14}	**6.7133E−02**	5.3793E−11	1.8950E−08	8.6382E−09	1.0962E−06	9.3771E−08	1.3371E−04
f_{15}	4.6427E−01	**8.1874E−01**	4.6427E−01	3.5137E−02	1.2493E−05	1.6955E−02	1.1058E−04
f_{16}	5.1436E−12	1.1364E−11	3.0199E−11	4.5043E−11	3.0199E−11	2.3985E−01	3.0199E−11
f_{17}	1.2118E−12	1.2118E−12	3.0199E−11	3.0199E−11	2.9450E−11	3.3384E−11	4.1997E−10
f_{18}	2.5839E−11	2.8827E−11	3.0199E−11	3.0199E−11	3.0199E−11	4.5043E−11	3.0199E−11
f_{19}	4.0806E−12	1.3369E−11	1.1023E−08	8.3520E−08	5.4941E−11	1.8500E−08	9.7555E−10
f_{20}	1.7634E−03	**2.6433E−01**	3.8249E−09	2.5974E−05	**1.0547E−01**	1.0188E−05	3.8202E−10
f_{21}	**3.7599E−01**	9.5912E−08	3.0199E−11	3.0199E−11	1.9527E−03	3.0199E−11	3.0199E−11
f_{22}	2.5831E−02	4.7193E−10	3.0199E−11	3.0199E−11	1.9527E−03	3.0199E−11	3.0199E−11
f_{23}	3.0199E−11	9.2008E−06	3.0199E−11	3.0199E−11	3.9881E−04	3.0199E−11	3.0199E−11
+	19	18	18	20	19	18	19
≈	3	3	2	3	3	3	1
-	1	2	3	0	1	2	3

p Values that are higher than 0.05 are denoted in bold face, whereas NaN indicates 'Not a Number' which is returned by the Wilcoxon test. The three symbols '+,' '≈' and '−' indicate that VPPSO performs significantly better, statistically similar and significantly worse in comparison with other algorithms, respectively

Table 19 Best results of the comparative algorithms for the welded beam design problem

Algorithm	x_1	x_2	x_3	x_4	Optimal cost
VPPSO	0.1961	3.3885	9.2006	0.1988	1.6740
CPSO [91]	0.202369	3.544214	9.04821	0.205723	1.72802
PSO [84]	0.2157	3.4704	9.0356	0.2658	1.85778
IPSO [92]	0.2444	6.2175	8.2915	0.2444	2.3810
MPA [93]	0.205728	3.470509	9.036624	0.205730	1.724853
GA [95]	0.2489	6.1730	8.1789	0.2533	2.4300
HGSO	0.2005	4.0017	8.6053	0.2410	1.9736
GWO [79]	0.205676	3.478377	9.03681	0.205778	1.72624
SSA	0.1880	3.5364	9.2523	0.1986	1.6880
WOA	0.1797	4.0355	9.8861	0.1958	1.8236
AOA[86]	0.2057	3.4705	9.0366	0.2057	1.7249
GSA [84]	0.2191	3.6661	10.000	0.2508	2.2291
HHO [86]	0.2134	3.5601	8.4629	0.2346	1.8561
EO [2]	0.2057	3.4705	9.03664	0.2057	1.7249

Table 20 Best results of the comparative algorithms for the speed reducer problem

Algorithm	x_1	x_2	x_3	x_4	x_5	x_6	x_7	Optimal weight
VPPSO	3.5000	0.7000	17.0007	7.3075	7.7340	3.3506	5.2867	2995
PSO [84]	3.500	0.70	17	7.74	7.85	3.36	5.389	2998.12
HHO [86]	3.4981	0.7	17	7.6398	7.8	3.3582	5.2853	2999.6
HGSO	3.6000	0.7148	17.0000	8.3000	8.3000	3.9000	5.5000	3433.0
GWO	3.5043	0.7000	17.0000	7.4386	7.7555	3.3606	5.2900	3002.9
SSA	3.5080	0.7000	17.0000	7.3386	7.8456	3.3568	5.2867	3002.4
WOA	3.5080	0.7000	17.0000	7.7490	7.8598	3.4031	5.2867	3018.5
GA [86]	3.5592	0.7133	19.659	7.9365	8.0197	3.6719	5.3276	3727.4
PSO [84]	3.500	0.70	17	7.74	7.85	3.36	5.389	2998.12
SCA [87]	3.508755	0.7	17	7.3	7.8	3.461020	5.289213	3030.563
GSA [88]	3.6	0.7	17	8.3	7.8	3.369658	5.289224	3051.12
AOA	3.5109	0.7	17	7.3	7.7198	3.3505	5.2867	2998.8

Table 21 Best results of the comparative algorithms for the pressure vessel design problem

Algorithm	x_1	x_2	x_3	x_4	Optimal cost
VPPSO	0.7783	0.3847	40.3274	199.9140	5886.1
CPSO [91]	0.8125	0.4375	42.091266	176.7465	6061.0777
PSO-DE [96]	0.8125	0.4375	42.098446	176.6366	6059.71433
HPSO [97]	0.8125	0.4375	42.0984	176.6366	6059.7143
GA [98]	0.81250	0.43750	42.097398	176.65405	6059.94634
HHO [86]	0.9833	0.4758	49.9297	98.9036	6391.9
GWO [79]	0.812500	0.434500	42.089181	176.758731	6051.5639
HGSO	1.1992	0.6511	61.8141	29.1838	7666.4
SSA	0.8031	0.3970	41.6104	184.2422	5962.7
WOA	1.0003	0.5510	51.3396	88.0599	6695.4
AOA	0.7831	0.3871	40.5777	196.4388	5893.9
SCA [86]	0.8951	0.4579	44.8371	147.3388	6403.7
ACO [99]	0.812500	0.437500	42.098353	176.637751	6059.7258

Table 22 Best results of the comparative algorithms for the tension/compression spring design problem

Algorithm	x_1	x_2	x_3	Optimal weight
VPPSO	0.0525	0.3756	10.2659	0.0127
PSO	0.0524	0.3746	10.3140	0.0127
GA [86]	0.0598	0.4121	9.1320	0.019824
HGSO	0.0500	0.3171	14.3710	0.0130
GWO	0.0513	0.3474	11.8763	0.0127
SSA	0.0527	0.3805	10.0417	0.0127
WOA	0.0538	0.4091	8.7700	0.0127
AOA	0.0529	0.3863	9.7450	0.0127
GSA [88]	0.05028	0.32368	13.52541	0.01270
SCA [86]	0.0500	0.3171	14.1417	0.012797
HHO [86]	0.0562	0.4754	6.6670	0.013016

Pressure vessel design problem

$$
\begin{aligned}
\min_{x}\quad & f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 \\
& 19.84x_1^2x_3 \\
\text{s.t.}\quad & g_1(x) = -x_1 + 0.0193x_3 \leq 0 \\
& g_2(x) = -x_2 + 0.00954x_3 \leq 0 \\
& g_3(x) = x_4 - 240 \leq 0 \\
& g_4(x) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0 \\
\text{range}\quad & 0 \leq x_i \leq 100, \quad i = 1, 2 \\
& 10 \leq x_i \leq 200, \quad i = 3, 4
\end{aligned}
$$

Tension/compression spring design problem

$$
\begin{aligned}
\min_{x}\quad & f(x) = x_1^2 x_2 (x_3 + 2) \\
\text{s.t.}\quad & g_1(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0 \\
& g_2(x) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \\
& g_3(x) = \frac{4x_2^2 - x_1x_2}{12566\left(x_2x_1^3 - x_1^4\right)} + \frac{1}{5108x_1^2} - 1 \leq 0 \\
& g_4(x) = 1 - \frac{140.45x_1}{x_2^2 x_3} \leq 0 \\
\text{range}\quad & 0.05 \leq x_1 \leq 2.00 \\
& 0.25 \leq x_2 \leq 1.30 \\
& 2.00 \leq x_3 \leq 15.00
\end{aligned}
$$

Data availability Data sharing not applicable because no datasets are analyzed or generated in this article.

Declaration

Competing interest The authors declare that they have no financial or personal relationships related to this work.

References

1. Shami TM, Grace D, Burr A, Mitchell PD (2022) Single candidate optimizer: a novel optimization algorithm. Evol Intell 1–25
2. Faramarzi A, Heidarinejad M, Stephens B, Mirjalili S (2020) Equilibrium optimizer: a novel optimization algorithm. Knowl-Based Syst 191:105190
3. Hashim FA, Houssein EH, Hussain K, Mabrouk MS, Al-Atabany W (2022) Honey badger algorithm: new metaheuristic algorithm for solving optimization problems. Math Comput Simul 192:84–110
4. Zervoudakis K, Tsafarakis S (2020) A mayfly optimization algorithm. Comput Ind Eng 145:106559
5. Rana N, Latiff MSA, Abdulhamid SM, Chiroma H (2020) Whale optimization algorithm: a systematic review of contemporary applications, modifications and developments. Neural Comput Appl 32(20):16245–16277
6. Shami TM, El-Saleh AA, Alswaitti M, Al-Tashi Q, Summakieh MA, Mirjalili S (2022) Particle swarm optimization: a comprehensive survey. IEEE Access
7. Ma S, Song S, Zhao J, Zhai L, Yang F (2020) Joint network selection and service placement based on particle swarm optimization for multi-access edge computing. IEEE Access 8:160871–160881
8. Shami TM, Grace D, Burr A, Vardakas JS (2019) Load balancing and control with interference mitigation in 5G heterogeneous networks. EURASIP J Wirel Commun Netw 2019(1):1–12
9. Al-Tashi Q, Akhir EAP, Abdulkadir SJ, Mirjalili S, Shami TM, Alhusssian H, Alqushaibi A, Alwadain A, Balogun AO, Al-Zidi N (2021) Classification of reservoir recovery factor for oil and gas reservoirs: a multi-objective feature selection approach. J Marine Sci Eng 9(8):888
10. Singh P, Chaudhury S, Panigrahi BK (2021) Hybrid MPSO-CNN: multi-level particle swarm optimized hyperparameters of convolutional neural network. Swarm Evol Comput 63:100863
11. Kaveh A, Zolghadr A (2014) Democratic pso for truss layout and size optimization with frequency constraints. Comput Struct 130:10–21
12. Chen Y, Yan J, Sareh P, Feng J (2020) Feasible prestress modes for cable-strut structures with multiple self-stress states using

particle swarm optimization. J Comput Civ Eng 34(3):04020003–04020003
13. Chen Y, Yan J, Feng J, Sareh P (2020) A hybrid symmetry-pso approach to finding the self-equilibrium configurations of prestressable pin-jointed assemblies. Acta Mech 231(4):1485–1501
14. Rahkar Farshi T, Ardabili AK (2021) A hybrid firefly and particle swarm optimization algorithm applied to multilevel image thresholding. Multim Syst 27(1):125–142
15. Chen Y, Yan J, Feng J, Sareh P (2021) Particle swarm optimization-based metaheuristic design generation of non-trivial flat-foldable origami tessellations with degreE−4 vertices. J Mech Design 143(1)
16. Jordehi AR (2015) Enhanced leader pso (elpso): a new PSO variant for solving global optimisation problems. Appl Soft Comput 26:401–417
17. Lai X, Zhou Y (2019) An adaptive parallel particle swarm optimization for numerical optimization problems. Neural Comput Appl 31(10):6449–6467
18. Zaman HRR, Gharehchopogh FS (2021) An improved particle swarm optimization with backtracking search optimization algorithm for solving continuous optimization problems. Eng Comput 1–35
19. Ezugwu AE, Agushaka JO, Abualigah L, Mirjalili S, Gandomi AH (2022) Prairie dog optimization algorithm. Neural Comput Appl 1–49
20. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
21. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of ICNN'95-international Conference on Neural Networks, vol 4, pp 1942–1948 . IEEE
22. Chi R, Su Y-x, Zhang D-h, Chi X-x, Zhang H-j (2019) A hybridization of cuckoo search and particle swarm optimization for solving optimization problems. Neural Comput Appl 31(1):653–670
23. Piotrowski AP, Napiorkowski JJ, Piotrowska AE (2020) Population size in particle swarm optimization. Swarm Evol Comput 58:100718
24. Nshimirimana R, Abraham A, Nothnagel G (2021) A multi-objective particle swarm for constraint and unconstrained problems. Neural Comput Appl 33(17):11355–11385
25. Isiet M, Gadala M (2019) Self-adapting control parameters in particle swarm optimization. Appl Soft Comput 83:105653
26. Eberhart R.C, Shi Y (2001) Tracking and optimizing dynamic systems with particle swarms. In: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), vol 1, pp 94–100. IEEE
27. Shi Y, Eberhart R.C (1999) Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary computation-CEC99 (Cat. No. 99TH8406), vol 3, pp 1945–1950. IEEE
28. Ghosh S, Das S, Kundu D, Suresh K, Panigrahi BK, Cui Z (2012) An inertia-adaptive particle swarm system with particle mobility factor for improved global optimization. Neural Comput Appl 21(2):237–250
29. Wang S, Liu G, Gao M, Cao S, Guo A, Wang J (2020) Heterogeneous comprehensive learning and dynamic multi-swarm particle swarm optimizer with two mutation operators. Inf Sci 540:175–201
30. Xia X, Gui L, Zhan Z-H (2018) A multi-swarm particle swarm optimization algorithm based on dynamical topology and purposeful detecting. Appl Soft Comput 67:126–140
31. Kao Y-T, Zahara E (2008) A hybrid genetic algorithm and particle swarm optimization for multimodal functions. Appl Soft Comput 8(2):849–857
32. Tang B, Xiang K, Pang M (2020) An integrated particle swarm optimization approach hybridizing a new self-adaptive particle swarm optimization with a modified differential evolution. Neural Comput Appl 32(9):4849–4883
33. Sedighizadeh D, Masehian E, Sedighizadeh M, Akbaripour H (2021) GEPSO: a new generalized particle swarm optimization algorithm. Math Comput Simul 179:194–212
34. Isiet M, Gadala M (2020) Sensitivity analysis of control parameters in particle swarm optimization. J Comput Sci 41:101086
35. Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), pp 69–73. IEEE
36. Yang X, Yuan J, Yuan J, Mao H (2007) A modified particle swarm optimizer with dynamic adaptation. Appl Math Comput 189(2):1205–1213
37. Panigrahi B, Pandi VR, Das S (2008) Adaptive particle swarm optimization approach for static and dynamic economic load dispatch. Energy Convers Manage 49(6):1407–1415
38. Nickabadi A, Ebadzadeh MM, Safabakhsh R (2011) A novel particle swarm optimization algorithm with adaptive inertia weight. Appl Soft Comput 11(4):3658–3670
39. Zhan Z-H, Zhang J, Li Y, Chung HS-H (2009) Adaptive particle swarm optimization. IEEE Trans Syst Man Cybern B (Cybernetics) 39(6):1362–1381
40. Jiao B, Lian Z, Gu X (2008) A dynamic inertia weight particle swarm optimization algorithm. Chaos Solitons Fractals 37(3):698–705
41. Fan S-KS, Chiu Y-Y (2007) A decreasing inertia weight particle swarm optimizer. Eng Optim 39(2):203–228
42. Chatterjee A, Siarry P (2006) Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. Comput Oper Res 33(3):859–871
43. Tang Y, Wang Z, Fang J-a (2011) Feedback learning particle swarm optimization. Appl Soft Comput 11(8):4713–4725
44. Li H.-R, Gao Y.-L (2009) Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation. In: 2009 second international conference on information and computing science, vol 1, pp 66–69. IEEE
45. Chen G, Huang X, Jia J, Min Z (2006) Natural exponential inertia weight strategy in particle swarm optimization. In: 2006 6th World Congress on Intelligent Control and Automation, vol. 1, pp. 3672–3675 . IEEE
46. Liu H, Zhang X-W, Tu L-P (2020) A modified particle swarm optimization using adaptive strategy. Expert Syst Appl 152:113353
47. Ratnaweera A, Halgamuge SK, Watson HC (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evol Comput 8(3):240–255
48. Xia X, Xing Y, Wei B, Zhang Y, Li X, Deng X, Gui L (2019) A fitness-based multi-role particle swarm optimization. Swarm Evol Comput 44:349–364
49. Ghasemi M, Akbari E, Rahimnejad A, Razavi SE, Ghavidel S, Li L (2019) Phasor particle swarm optimization: a simple and efficient variant of pso. Soft Comput 23(19):9701–9718
50. Van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239
51. Gou J, Lei Y-X, Guo W-P, Wang C, Cai Y-Q, Luo W (2017) A novel improved particle swarm optimization algorithm based on individual difference evolution. Appl Soft Comput 57:468–481
52. Ye W, Feng W, Fan S (2017) A novel multi-swarm particle swarm optimization with dynamic learning strategy. Appl Soft Comput 61:832–843

53. Chen Y, Li L, Peng H, Xiao J, Yang Y, Shi Y (2017) Particle swarm optimizer with two differential mutation. Appl Soft Comput 61:314–330
54. Li W, Meng X, Huang Y, Fu Z-H (2020) Multipopulation cooperative particle swarm optimization with a mixed mutation strategy. Inf Sci 529:179–196
55. Karim AA, Isa NAM, Lim WH (2021) Hovering swarm particle swarm optimization. IEEE Access 9:115719–115749
56. Xia X, Tang Y, Wei B, Gui L (2019) Dynamic multi-swarm particle swarm optimization based on elite learning. IEEE Access 7:184849–184865
57. Kıran MS, Gündüz M, Baykan ÖK (2012) A novel hybrid algorithm based on particle swarm and ant colony optimization for finding the global minimum. Appl Math Comput 219(4):1515–1521
58. Pan X, Xue L, Lu Y, Sun N (2019) Hybrid particle swarm optimization with simulated annealing. Multim Tools Appl 78(21):29921–29936
59. Zhang X, Lin Q, Mao W, Liu S, Dou Z, Liu G (2021) Hybrid particle swarm and grey wolf optimizer and its application to clustering optimization. Appl Soft Comput 101:107061
60. Xia X, Gui L, He G, Xie C, Wei B, Xing Y, Wu R, Tang Y (2018) A hybrid optimizer based on firefly algorithm and particle swarm optimization algorithm. J Comput Sci 26:488–500
61. Laskar NM, Guha K, Chatterjee I, Chanda S, Baishnab KL, Paul PK (2019) Hwpso: A new hybrid whale-particle swarm optimization algorithm and its application in electronic design optimization problems. Appl Intell 49(1):265–291
62. Mendes R, Kennedy J, Neves J (2004) The fully informed particle swarm: simpler, maybe better. IEEE Trans Evol Comput 8(3):204–210
63. Zeng N, Wang Z, Liu W, Zhang H, Hone K, Liu X (2020) A dynamic neighborhood-based switching particle swarm optimization algorithm. IEEE Trans Cybern
64. Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 10(3):281–295
65. Parsopoulos KE, Vrahatis MN (2019) Upso: a unified particle swarm optimization scheme. In: International conference of computational methods in sciences and engineering 2004 (ICCMSE 2004), pp 868–873. CRC Press
66. Qu B-Y, Suganthan PN, Das S (2012) A distance-based locally informed particle swarm model for multimodal optimization. IEEE Trans Evol Comput 17(3):387–402
67. Liu Z, Nishi T (2022) Strategy dynamics particle swarm optimizer. Inf Sci 582:665–703
68. Zhang X, Wang X, Kang Q, Cheng J (2019) Differential mutation and novel social learning particle swarm optimization algorithm. Inf Sci 480:109–129
69. Kohler M, Vellasco MM, Tanscheit R (2019) PSO+: a new particle swarm optimization algorithm for constrained problems. Appl Soft Comput 85:105865
70. Chen K, Xue B, Zhang M, Zhou F (2020) Novel chaotic grouping particle swarm optimization with a dynamic regrouping strategy for solving numerical optimization tasks. Knowl-Based Syst 194:105568
71. Machado JT, Pahnehkolaei SMA, Alfi A (2021) Complex-order particle swarm optimization. Commun Nonlinear Sci Numer Simul 92:105448
72. Xia X, Gui L, He G, Wei B, Zhang Y, Yu F, Wu H, Zhan Z-H (2020) An expanded particle swarm optimization based on multi-exemplar and forgetting ability. Inf Sci 508:105–120
73. Xu G, Cui Q, Shi X, Ge H, Zhan Z-H, Lee HP, Liang Y, Tai R, Wu C (2019) Particle swarm optimization based on dimensional learning strategy. Swarm Evol Comput 45:33–51
74. Zhang Y, Liu X, Bao F, Chi J, Zhang C, Liu P (2020) Particle swarm optimization with adaptive learning strategy. Knowl-Based Syst 196:105789
75. Xia X, Gui L, Yu F, Wu H, Wei B, Zhang Y-L, Zhan Z-H (2019) Triple archives particle swarm optimization. IEEE Trans Cybern 50(12):4862–4875
76. Li T, Shi J, Deng W, Hu Z (2022) Pyramid particle swarm optimization with novel strategies of competition and cooperation. Appl Soft Comput 121:108731
77. Chen X, Tianfield H, Du W (2021) Bee-foraging learning particle swarm optimization. Appl Soft Comput 102:107134
78. Wang R, Hao K, Chen L, Wang T, Jiang C (2021) A novel hybrid particle swarm optimization using adaptive strategy. Inf Sci 579:231–250
79. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
80. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102
81. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
82. Mirjalili S, Lewis A (2013) S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm Evol Comput 9:1–14
83. Digalakis JG, Margaritis KG (2001) On benchmarking functions for genetic algorithms. Int J Comput Math 77(4):481–506
84. Hashim FA, Houssein EH, Mabrouk MS, Al-Atabany W, Mirjalili S (2019) Henry gas solubility optimization: a novel physics-based algorithm. Futur Gener Comput Syst 101:646–667
85. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
86. Hashim FA, Hussain K, Houssein EH, Mabrouk MS, Al-Atabany W (2021) Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. Appl Intell 51(3):1531–1551
87. Mirjalili S (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133
88. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
89. Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
90. He X, Zhou Y (2018) Enhancing the performance of differential evolution with covariance matrix self-adaptation. Appl Soft Comput 64:227–243
91. He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 20(1):89–99
92. He S, Prempain E, Wu Q (2004) An improved particle swarm optimizer for mechanical design optimization problems. Eng Optim 36(5):585–605
93. Faramarzi A, Heidarinejad M, Mirjalili S, Gandomi AH (2020) Marine predators algorithm: a nature-inspired metaheuristic. Expert Syst Appl 152:113377
94. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Futur Gener Comput Syst 97:849–872
95. Deb K (1991) Optimal design of a welded beam via genetic algorithms. AIAA J 29(11):2013–2015
96. Liu H, Cai Z, Wang Y (2010) Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl Soft Comput 10(2):629–640

97. He Q, Wang L (2007) A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Appl Math Comput 186(2):1407–1422
98. Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41(2):113–127
99. Kaveh A, Talatahari S (2010) An improved ant colony optimization for constrained engineering design problems. Eng Comput
100. Duan C, Deng C, Gharaei A, Wu J, Wang B (2018) Selective maintenance scheduling under stochastic maintenance quality with multiple maintenance actions. Int J Prod Res 56(23):7160–7178
101. Alswaitti M, Albughdadi M, Isa NAM (2018) Density-based particle swarm optimization algorithm for data clustering. Expert Syst Appl 91:170–186
102. Gharaei A, Hoseini Shekarabi SA, Karimi M (2020) Modelling and optimal lot-sizing of the replenishments in constrained, multi-product and bi-objective EPQ models with defective products: Generalised cross decomposition. Int J Syst Sci: Oper Logist 7(3):262–274
103. Gharaei A, Karimi M, Hoseini Shekarabi SA (2020) Joint economic lot-sizing in multi-product multi-level integrated supply chains: generalized benders decomposition. Int J Syst Sci: Oper Logist 7(4):309–325
104. Abdel-Basset M, Chang V, Mohamed R (2021) A novel equilibrium optimization algorithm for multi-thresholding image segmentation problems. Neural Comput Appl 33(17):10685–10718

Authors and Affiliations

Tareq M. Shami[1] · Seyedali Mirjalili[2,3] · Yasser Al-Eryani[4] · Khadija Daoudi[5] · Saadat Izadi[6] · Laith Abualigah[7,8,9,10,11]

✉ Tareq M. Shami
tareq.al-shami@york.ac.uk

[1] Department of Electronic Engineering, University of York, Heslington, York YO10 5DD, UK

[2] Centre for Artificial Intelligence Research and Optimisation, Torrens University Australia, Brisbane, Australia

[3] Yonsei Frontier Lab, Yonsei University, Seoul, South Korea

[4] Ericsson Canada Inc, 349 Terry Fox Dr, Kanata, ON K2K 2V6, Canada

[5] Department of Electronics, National institute of poste and telecommunication, Av. Allal Al Fassi, Rabat, Morocco

[6] Department of Computer Engineering and Information Technology, Razi University, Kermanshah, Iran

[7] Prince Hussein Bin Abdullah College for Information Technology, Al Al-Bayt University, Mafraq 130040, Jordan

[8] Hourani Center for Applied Scientific Research, Al-Ahliyya Amman University, Amman 19328, Jordan

[9] Faculty of Information Technology, Middle East University, Amman 11831, Jordan

[10] Faculty of Information Technology, Applied Science Private University, Amman 11931, Jordan

[11] School of Computer Sciences, Universiti Sains Malaysia, Pulau Pinang 11800, Malaysia