Expert Systems With Applications 140 (2020) 112882

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A constrained multi-swarm particle swarm optimization without velocity for constrained optimization problems

Koon Meng Ang[a], Wei Hong Lim[a,*], Nor Ashidi Mat Isa[b], Sew Sun Tiang[a], Chin Hong Wong[a]

[a] Faculty of Engineering, Technology and Built Environment, UCSI University, Kuala Lumpur 56000, Malaysia
[b] School of Electrical and Electronic Engineering, Universiti Sains Malaysia, Nibong Tebal 14300, Malaysia

ARTICLE INFO

Article history:
Received 25 May 2019
Revised 28 July 2019
Accepted 17 August 2019
Available online 18 August 2019

Keywords:
Constrained optimization
Particle swarm optimization
Current swarm evolution
Memory swarm evolution

ABSTRACT

The original particle swarm optimization (PSO) is not able to tackle constrained optimization problems (COPs) due to the absence of constraint handling techniques. Furthermore, most existing PSO variants can only perform well in certain types of optimization problem and tend to suffer with premature convergence due to the limited search operator and directional information used to guide the search process. An improved PSO variant known as the constrained multi-swarm particle swarm optimization without velocity (CMPSOWV) is proposed in this paper to overcome the aforementioned drawbacks. Particularly, a constraint handling technique is first incorporated into CMPSOWV to guide population searching towards the feasible regions of search space before optimizing the objective function within the feasible regions. Two evolution phases known as the current swarm evolution and memory swarm evolution are also introduced to offer the multiple search operators for each CMPSOWV particle, aiming to improve the robustness of algorithm in solving different types of COPs. Finally, two diversity maintenance schemes of multi-swarm technique and probabilistic mutation operator are incorporated to prevent the premature convergence of CMPSOWV. The overall optimization performances of CMPSOWV in solving the CEC 2006 and CEC 2017 benchmark functions and real-world engineering design problems are compared with selected constrained optimization algorithms. Extensive simulation results report that the proposed CMPSOWV has demonstrated the best search accuracy among all compared methods in solving majority of problems.

1. Introduction

The field of optimization has received significant attention in recent years as a promising tool for decision making. Depending on the objective function used to describe a specific goal to be achieved by an optimization problem, the optimal combination of decision variables obtained can either lead to the smallest objective function value for minimization problems or the largest objective function value for maximization problems. Majority of the real-world engineering application such as product development are considered as the constrained optimization problems (COPs). The objective functions used to describe the preliminary design model of product are generally represented using a set of analytical equations, while the product specifications are formulated as technical constraints. The presence of optimization constraints tend to reduce the feasible regions of search space, resulting in the COPs become more difficult to solve as compared to the unconstrained counterparts (Mezura-Montes & Coello Coello, 2011; Michalewicz & Schoenauer, 1996; Runarsson & Xin, 2000). In order to solve the COPs successfully, the optimal set of decision variables obtained not only need to optimize the objective functions, but also to satisfy all technical constraints (Mezura-Montes & Coello Coello, 2011; Michalewicz & Schoenauer, 1996; Runarsson & Xin, 2000).

Conventionally, deterministic methods such as branch-and-bound (Lawler & Wood, 1966), multi-dimensional bisection (Wood, 1991) and interval arithmetic (Benhamou & Older, 1997) are applied to solve COPs by exploiting the underlying mathematical structure of given problems. These deterministic methods require an explicit mathematical expression of the optimization problem and good guess of initial point in order to locate the global optimum solution of problem effectively. Nevertheless,

* Corresponding author.
E-mail addresses: 1001436889@student.ucsiuniversity.edu.my (K.M. Ang), limwh@ucsiuniversity.edu.my (W.H. Lim), ashidi@usm.my (N.A.M. Isa), tiangss@ucsiuniversity.edu.my (S.S. Tiang), wongch@ucsicollege.edu.my (C.H. Wong).

https://doi.org/10.1016/j.eswa.2019.112882
0957-4174/

most optimization models of real-world applications are black box functions that are represented implicitly and information such as number of local minima are usually unknown in priori. These undesirable features make deterministic methods become the less preferred option for practitioners to solve the real world COPs. On the other hand, evolutionary algorithms (EAs) and swarm intelligence (SI) algorithms emerge as the effective solutions to solve optimization problems by leveraging the intelligent search mechanisms inspired by physical processes, natural evolution or collective behaviors of animals to create the enhanced population in the next generation. Unlike their deterministic counterpart, both of EA and SI algorithm approaches have competitive advantages such as problem independent, minimal adherence to mathematical basis and simplicity in implementation. Most EAs and SI algorithms are primarily designed to solve unconstrained optimization problems. Different constraint handling methods are incorporated into the original EAs or SI algorithms to guide the population approaching the feasible regions of search space promptly before locating the global optimum within the feasible regions (Mezura-Montes & Coello Coello, 2011).

Particle swarm optimization (PSO) is a popular SI algorithm used to solve the global optimization in continuous search space (Banks, Vincent & Anyakoha, 2007, 2008; Kennedy & Eberhart, 1995; Valle, Venayagamoorthy, Mohagheghi, Hernandez & Harley, 2008). It has an intelligent search mechanism inspired from the social behavior of bird flocking and fish schooling in searching for food sources. Each particle represents a candidate solution in search space, while the food sources refers to the global optimum of problem. The search trajectory of each PSO particle can be adjusted based on the best experience achieved by itself and the entire population during the optimization process. Due to its simplicity in implementation and promising convergence characteristic, PSO has been successfully applied to solve different types of real-world optimization problems (Mistry, Zhang, Neoh, Lim & Fielding, 2017; Mohanty, Mahapatra & Singh, 2016; Solihin, Wahyudi & Akmeliawati, 2009; Tang, Ang, Ariffin & Mashohor, 2014, Ang; Van & Kang, 2016; Yao, Damiran & Lim, 2017; Yao, Lai, & Lim, 2015; Zhao, Wen, Dong, Xue & Wong, 2012). Furthermore, substantial amounts of PSO variants with enhanced search performance were introduced in past decades via the modification of parameter adaptation, population topologies, learning strategies and etc. (Banks et al., 2007, 2008; Kennedy & Eberhart, 1995; Lan, Zhang, Tang, Liu & Luo, 2019; Lim & Isa, 2015; Lim et al., 2018; Liu, Cui, Lu, Liu & Deng, 2019; Valle et al., 2008; Xu et al., 2019).

A PSO variant known as particle swarm optimization without velocity (PSOWV) was proposed to enhance the convergence speed of algorithm (El-Sherbiny, 2011). A drastic modification was introduced in PSOWV by discarding the velocity component of particles and the new position of particle was updated based on the linear combination of its personal best position and global best position. Although the convergence speed of PSOWV in solving selected benchmark problems was improved, its capability in tackling the more complex optimization problems such as COPs remain arduous. Firstly, the PSOWV has an inherent drawback similar with the original PSO and most of its variants for not being able to solve COPs because it is not equipped with any constraint handling techniques. Secondly, PSOWV has only one single search operator with limited exploration and exploitation strengths and its excellent search performance is restricted in certain categories of test problems while producing poor results in solving other types of problems (Li, Yang & Nguyen, 2012; Lim et al., 2018; Vrugt, Robinson & Hyman, 2009; Wang et al., 2011). Finally, the search operator of PSOWV relies on both personal best position and global best position in guiding the search process of particles. If the global best position is trapped into the local optimal, the remaining particles tend to be misguided and converge towards the inferior of search space, leading to the premature convergence of algorithm (Banks et al., 2007, 2008; Kennedy & Eberhart, 1995; Valle et al., 2008).

An improved version of algorithm known as constrained multi-swarm particle swarm optimization without velocity (CMPSOWV) is proposed in this paper to overcome the shortcomings of PSOWV. The main contributions of the proposed CMPSOWV are summarized as follows:

- A constraint handling technique is incorporated into CMPSOWV to guide population searching towards the feasible regions of search space before optimizing the objective functions within the feasible regions.
- Two evolution phases known as current swarm evolution and memory swarm evolution are introduced into CMPSOWV to enable the particles performing search processes with different exploration and exploitation strength based on multiple search operators.
- A multi-swarm technique and a mutation scheme are also incorporated into CMPSOWV to prevent the population diversity loss and premature convergence of algorithm during the search process.
- The optimization performances of CMPSOWV are evaluated rigorously with the CEC 2006 and CEC 2017 benchmark functions and four engineering design problems.

The remaining sections of this paper are summarized as follows. The related works of this research are presented in Section 2, while the detailed description of CMPSOWV are provided in Section 3. Extensive simulation studies used to evaluate the performance of CMPSOWV in solving COPs are described in Section 4. Finally, the research findings and future works are this research are concluded in Section 5.

2. Related works

2.1. Constrained optimization problems

The constrained optimization problems (COPs) in real-world applications aim to maximize or minimize the given objective functions. Without loss of generality, constrained minimization problems are considered throughout this paper. Assume that Ω is the feasible region of search space and S is a D-dimensional rectangular space of $\Re^D$, where $\Omega \subseteq S$. Denote $X = [X_1,..., X_d,..., X_D]$ as the feasible solution vector, where $X \in \Omega \subseteq S$. Suppose thatX_d refers to the d-th dimensional component of a feasible solution vector X and it must fulfil the upper and lower boundary limits denoted as X_d^U and X_d^L, respectively, where $X_d^L \le X_d \le X_d^U$. Define$f(X)$as the objective function value of solution X, while$g_j(X)$ and $h_j(X)$ be the values of the j-th equality constraint and inequality constraint, respectively. Therefore, the mathematical formulation of a COP can be expressed as follows (Efrén Mezura-Montes & Coello Coello, 2011; Runarsson & Xin, 2000):

$$\min_{X=[X_1,...,X_d,...,X_D]\in\Re^D} f(X)$$
$$\text{Subject to} \begin{cases} g_j(X) \le 0, \ j = 1, \ldots, G \\ h_j(X) = 0, j = G+1, ..., H \end{cases} \quad (1)$$

where G and $(H-G)$represent the numbers of inequality and equality constraints of a COP, respectively.

For the sake of convenience, each of the j-th equality constraint of COP in Eq. (1) can be transformed into an equivalent inequality form denoted as (Efrén Mezura-Montes & Coello Coello, 2011; Runarsson & Xin, 2000):

$$\left|h_j(x)\right| - \delta \ \le 0, \ j = G+1, ..., H \quad (2)$$

where δ is a positive tolerance parameter for the equality constraints. Denote $\varphi_j(X)$ as the degree of constraint violation of a solution vector X on each j-th inequality or equality constraint, then:

$$\varphi_j(X) = \begin{cases} \max\{0,\ g_j(x)\}, & 1 \le j \le G \\ \max\{0,\ |h_j(x)| - \delta\}, & G+1 \le j \le H \end{cases} \quad (3)$$

The total degree of constraint violation of solution vector X can be obtained by summing up the individual violation due to each inequality and equality constraints. Therefore,

$$\varphi(X) = \sum_{j=1}^{H} \varphi_j(X) \quad (4)$$

2.2. Particle swarm optimization

Suppose that the population size of PSO swarm and the dimensional size of a given problem are represented as N and D, respectively. Denote $V_i = [V_{i,1}, ..., V_{i,d}, ..., V_{i,D}]$ and $X_i = [X_{i,1}, ..., X_{i,d}, ..., X_{i,D}]$ as the velocity and position of each i-th particle in the search space, respectively, where $i = 1, ..., N$ and $d = 1, ..., D$. Each of the i-th particle is able to memorize its best searching performance achieved so far and this personal best positon is represented as $P_{best,i} = [P_{best,i,1}, ..., P_{best,i,d}, ..., P_{best,i,D}]$. Meanwhile, the global best position is the best experience achieved by the population so far and it is denoted as $G_{best} = [G_{best,1}, ..., G_{best,d}, ..., G_{best,D}]$. The collaborative behavior of PSO swarm during the search process is emulated by stochastically adjusting the velocity of each i-th particle based on its self-cognitive experience $P_{best,i}$ and the social experience G_{best}. The new position of each particle is then obtained based on the updated velocity vector.

At the $(t+1)$-th iteration of search process, the d-th dimension of velocity and position of each i-th particle, denoted as$V_{i,d}(t+1)$ and $X_{i,d}(t+1)$, respectively, are updated as (Banks et al., 2007, 2008; Kennedy & Eberhart, 1995; Valle et al., 2008):

$$V_{i,d}(t+1) = \omega V_{i,d}(t) + c_1 r_1 \left(P_{best,i,d} - X_{i,d}(t)\right) + c_2 r_2 \left(G_{best,d} - X_{i,d}(t)\right) \quad (5)$$

$$X_{i,d}(t+1) = X_{i,d}(t) + V_{i,d}(t+1) \quad (6)$$

where ω is an inertia weight used to determine how much the previous velocity of a particle is preserved; $c_1$andc_2 are the acceleration coefficients used to control the influence of self-cognitive (i.e., $P_{best,i}$) and social (i.e., G_{best}) component of particle; r_1 and r_2 are two random numbers generated from uniform distribution in the range of 0 to 1, where r_1,$r_2 \in [0, 1]$. The objective function value of the updated position of particle is evaluated and compared with those of personal best positon and global best position. The latter two positions are replaced if the former one has better fitness. The searching process of PSO is repeated until all termination conditions are met.

2.3. Particle swarm optimization without velocity

An improved PSO variant known as particle swarm optimization without velocity (PSOWV) was proposed to enhance the convergence speed of algorithm by discarding the velocity component of particle (El-Sherbiny, 2011). For each i-th PSOWV particle, the d-th component of the position vector in every $(t+1)$-th iteration is determined based on the random linear combination between its personal best position and the global best position (El-Sherbiny, 2011):

$$X_{i,d}(t+1) = c_1 r_1 P_{best,i,d}(t) + c_2 r_2 G_{Best,d}(t) \quad (7)$$

Similar fitness evaluation and performance comparison are conducted between the updated position, personal best position and global best position of particle. While the convergence speed of PSOWV in solving certain tested problems has shown improvement, the capability of PSOWV in solving more challenging problems such as COPs remain arduous. It is also observed that the PSOWV might only perform well in certain category of problem and susceptible to premature convergence due to the employment of single search operator that relies on the information of personal best position and global best position.

2.4. Other variants of constrained PSO

Different variants of constrained PSO were proposed in past decades to solve COPs more effectively. A modified PSO known as particle evolutionary swarm optimization (PESO) with the enhanced population diversity was proposed by adding two perturbation operators to overcome the premature convergence issue (Zavala, Aguirre & Diharce, 2005). An improved version of PESO was proposed by introducing a new ring neighborhood structure and a new constraint handling technique based on the feasibility rule and sum of constraint violation (Hernandez Aguirre, Muñoz Zavala & Villa Diharce, 2007). A co-evolutionary particle swarm optimization (CPSO) was proposed by He and Wang (2007) to solve COPs by employing two types of swarms simultaneously. Particularly, the first type of multiple swarms of CPSO was used to search for optimum decision variables, while the second type of single swarm was used to adjust the penalty factors adaptively during fitness function evaluation. A modified CPSO was proposed by Krohling and Coelho (2006) to generate the acceleration coefficients of each particle with Gaussian distribution. A ranking selection-based particle swarm optimization (RSPSO) was proposed by Wang and Yin (2008))) to solve various engineering design problems with mixed variables. A ranking selection scheme was incorporated into RSPSO to control the exploration and exploitation strengths of particles in different stages of search processes. A hybrid PSO variant known as constrained particle swarm optimization with shake mechanism (CPSO-Shake) was proposed to alleviate the premature convergence issue with bi-population technique and shake mechanism (Cagnina, Esquivel & Coello, 2011). The division of an entire CPSO-Shake population into two subpopulations allow each group of particles exploring different regions of search space simultaneously, while the shake mechanism was applied to guide the good particle moving towards a random direction when more than 10% of population members were infeasible.

An improved particle swarm optimization (IPSO) was incorporated with a new parameter adaptation strategy and a new constraint handling technique to solve COPs without requiring any extra parameters or user intervention (Efrén Mezura-Montes & Flores-Mendoza, 2009). An improved vector particle swarm optimization (IVPSO) was proposed to solve COPs by employing the constraint preserving method to handle the constraint conflicts (Sun, Zeng & Pan, 2011). A “fly-back” mechanism was applied to pull each escaped IVPSO back to their original feasible regions and a multi-dimensional search approach was adopted to search the new feasible position of each IVPSO particle when it escaped from the feasible regions. A modified barebones particle swarm optimization (OBPSO) was prosed to solve nonlinear COPs by incorporating the opposition-based learning into the barebones particle swarm optimization (Wang, 2012). A novel boundary search strategy was employed by OBPSO to approach the boundary between the feasible and infeasible regions of search space, while an adaptive penalty method was proposed for constraint handling purpose. A cultural-based constrained particle swarm optimization (C-CPSO) was proposed as a novel heuristic to solve COPs by incorporating the information of objective function and constraint violation into four sections of belief space known as normative knowledge, spatial knowledge, situational knowledge and temporal knowledge (Daneshyari & Yen, 2012). The knowledge available in belief space enabled the interswarm communication that facilitated the selection of leading particles at the personal, swarm and global levels.

A constraint violation with interval arithmetic particle swarm optimization (CVI-PSO) was proposed to convert the COPs into unconstrained bi-objective optimization problems and solved with the lexicographic method (Mazhoud, Hadj-Hamou, Bigeon & Joyeux, 2013). The second objective of CVI-PSO was the total constraint violation function evaluated using internal arithmetic by considering the distance of each position of particle to the feasible regions. Several modification was introduced into the self-adaptive mix of particle swarm optimization (SAM-PSO) to enhance its search performance in different types of COPs (Elsayed, Sarker & Mezura-Montes, 2014). A self-adaptive parameter control strategy was first incorporated into SAM-PSO to fine tune the constriction factor and acceleration coefficient of each particle based on its search performance. An ensemble of different PSO variants was also included in SAM-PSO to balance the exploration and exploitation searches by assigning the better performing PSO variants with more particles based on the improvement index.

The Nelder-Mead (NM) simplex search method was hybridized with PSO to form NM-PSO and solve COPs by leveraging the high efficiency of NM simple search method in local search and good performance of PSO in global search, respectively (Zahara & Kao, 2009). The adoption of NM simple search method was proven able to reduce the computational time of NM-PSO. An improved accelerated particle swarm optimization (IAPSO) was proposed to handle the COPs with different types of design variables by leveraging the memory of particle and swarm to guide each particle discovering the promising regions of search space (Ben Guedria, 2016). The integrated control functions of IAPSO was also fine-tuned to achieve proper balancing of exploration and exploitation searches of algorithm. A novel hybrid particle swarm optimization with interval analysis method ([I]PSO) was proposed by incorporating a set inverter via interval analysis algorithm to remove the invalid region of search space (Machado-Coelho et al., 2017). A space cleaning algorithm was also included in [I]PSO to reduce the complexity of algorithm by eliminating the intervals without optimum values. A parallel boundary search particle swarm optimization (PBPSO) was introduced to reinforce the optimization capability of PSO through a cooperative mechanism (Liu, Li, Zhu & Chen, 2018). Particularly, a modified PSO was used to perform global search in one branch while the local search in another branch was conducted using the subset constraint boundary narrower function and sequential quadratic programming method.

A hybrid constrained optimization algorithm that combined both PSO and differential evolution (i.e., PSO-DE) was proposed to solve COPs by leveraging the good exploration search of DE algorithm in addressing the premature convergence issue (Liu, Cai & Wang, 2010). A similar PSO variant was applied to identify the optimal parameters of seismic isolators (Quaranta, Marano, Greco & Monti, 2014). An improved quantum-behaved particle swarm optimization consisted of a mutation operator with Gaussian probability distribution (G-QPSO) was developed (Coelho, 2010). In particular, the Gaussian mutation operator was used in G-QPSO to prevent the stagnation of particles in the local optima regions of search space. Another hybrid algorithm that combined an improved quantum-behave particle swarm optimization with the simplex algorithm (IQPSOS) was developed to solve a real-world engineering application known as load flow optimization problem (Davoodi, Hagh & Zadeh, 2014). Notably, the excellent local search capability of simplex algorithm was utilized by IQPSOS to fine tune the solutions obtained earlier. A hybrid algorithm consists of PSO, genetic algorithm (GA) and ant colony optimization (ACO) was proposed to improve the exploration and exploitation search capability of algorithm (Tam, Ong, Ismail, Ang & Khoo, 2019). Particularly, ACO was employed to enhance the exploration strength of GA through standard mutation to avoid the local entrapment, while PSO contributed in exploitation search through a refined mutation operator.

3. Constrained multi-swarm particle swarm optimization without velocity

In this section, the constraint handling method incorporated into the proposed CMPSOWV is first described. This is followed by the search mechanisms of both current swarm evolution and memory swarm evolution, as well as the diversity maintenance schemes incorporated. Finally, the overall framework of CMPSOWV is summarized.

3.1. Constraint handling method

The proposed CMPSOWV is incorporated with a constraint handling method to ensure the optimal solutions found not only can optimize the objective function, but also satisfy all functional constraints. Penalty method is a popular constraint handling methods used to penalize the overall fitness of an infeasible solution (Mezura-Montes & Coello Coello, 2011). The overall fitness of an infeasible particle is obtained by adding the original fitness function value with the total constraint violation, where the latter value can be obtained by multiplying the degree of constraint violation with a penalty factor. A major challenge of employing penalty method to solve COP is to determine a suitable penalty factor for different types of COPs so that the infeasible solution is not under- or over-penalized (Mezura-Montes & Coello Coello, 2011).

On the other hand, Deb's rule is a more intuitive approach in comparing the feasibility of two solutions without requiring to determine any parameters such as penalty factor (Deb, 2000; Mezura-Montes & Coello Coello, 2011). Hence, it is incorporated as the constraint handling method for CMPSOWV. Suppose that the i-th CMPSOWV particle with current position of X_i has the fitness and total constraint violation of $f(X_i)$ and $\varphi(X_i)$ respectively. Meanwhile, $f(X_i)$ and $\varphi(X_i)$ represent the fitness and total constraint violation of the j-th particle, respectively. For a given constrained minimization problem, the feasibility of both solutions, (i.e., X_i and X_j) are compared with Deb's rule based on the following guidelines (Deb, 2000; Mezura-Montes & Coello Coello, 2011): (a) the i-th feasible particle with $\varphi(X_i) \leq 0$ is better than the j-th infeasible particle with $\varphi(X_i) > 0$, (b) among two particles with feasible solutions where $\varphi(X_i)$, $\varphi(X_j) \leq 0$, the i-th particle is better than the j-th particle if $f(X_i) < f(X_j)$ and (c) among two particles with infeasible solutions where $\varphi(X_i)$, $\varphi(X_j) > 0$, the i-th particle is better than the j-th particle if $\varphi(X_i) < \varphi(X_j)$.

The pseudocode used to compare the current solution X_i and personal best position $P_{best,i}$ of the i-th CMPSOWV particle based on their respective fitness and total constraint violation values are presented in Fig. 1. The same pseudocode can be used to compare the feasibility between solutions X_i and global best position G_{best}. Deb's rule is then used to determine if the new position $X_i$generated by the i-th particle can replace $P_{best,i}$ and G_{best}. By incorporating with the Deb's rule, CMPSOWV is able to guide the population to search from infeasible regions towards feasible regions, followed by the optimization of objective function within the feasible regions.

3.2. Current swarm evolution

The concept of multiple search operator is incorporated into CMPSOWV via two evolution phases known as the current swarm evolution and memory swarm evolution, aiming to improve the robustness of proposed algorithm in handling different types of COPs. The current swarm evolution of CMPSOWV is essentially an

Algorithm 1: Deb's Rule

Input: $X_i, f(X_i), \varphi(X_i), P_{best,i}, f(P_{best,i}), \varphi(P_{best,i})$

01: **If** $\varphi(X_i) \le 0$ **and** $\varphi(P_{best,i}) \le 0$ **then** /*Check if both solutions are feasible*/

02: **If** $f(X_i) < f(P_{best,i})$ **then** /* Compare the fitness if both solutions are feasible */

03: $P_{best,i} \leftarrow X_i,\ f(P_{best,i}) \leftarrow f(X_i),\ \varphi(P_{best,i}) \leftarrow \varphi(X_i)$;

04: **End If**

05: **Else If** $\varphi(X_i) < \varphi(P_{best,i})$ **then** /* Compare the total constraint violation if any solution is feasible*/

06: $P_{best,i} \leftarrow X_i,\ f(P_{best,i}) \leftarrow f(X_i),\ \varphi(P_{best,i}) \leftarrow \varphi(X_i)$;

07: **End If**

Output: $P_{best,i}, f(P_{best,i}), \varphi(P_{best,i})$

Fig. 1. The pseudocode of Deb's rule used to update the personal best position of *i*-th particle and the associated fitness and total constraint violation.

Algorithm 2: Formation of Multi-Swarm

Input: $\mathbf{P} = [X_1,...,X_i,...X_N], K, N^k, N$

01: $k = 1$; /*Initialize the sub-swarm index*/

02: **While** $\mathbf{P} \neq \varnothing$ **do** /*If the main population is not empty*/

03: Randomly generate U^k;

04: Identify the *i*-th particle with $P_{best,i}$ that is nearest to U^k;

05: Combine the remaining $N^k - 1$ particles that are nearest to $P_{best,i}$ to form $\mathbf{P}^{sub,k}$;

06: Eliminate the sub-swarm members in $\mathbf{P}^{sub,k}$ from $\mathbf{P}$;

07: $k \leftarrow k + 1$;

08: **End While**

Output: K sub-swarms of $\mathbf{P}^{sub,k} = [X_1^k,...,X_i^k,...X_{N^k}^k]$, where $k = 1,...,K$

Fig. 2. The pseudocode of dividing the main population of CMPSOWV into *K* sub-swarms.

improved version of the search operator of PSOWV used to update the current position of particle. For the original PSOWV, all particles are guided by the similar direction information by referring to their personal best position and global best position. While this search operator is able to enhance the convergence speed of PSOWV in certain types of problem, the global best particle tends to be trapped into the local optima regions of more complicated problems. The remaining PSOWV particles are then misguided and converge towards the inferior regions of search space, leading to the diversity loss of population and premature convergence of algorithm.

In order to overcome the drawbacks of PSOVW, some modifications are proposed into the current swarm evolution of CMPSOWV in updating the new position of particles. A multi-swarm approach is introduced into the current swarm evolution of CMPSO, aiming to maintain the diversity of population by allowing each particle to perform searching based on different direction information. Denote $\mathbf{P} = [X_1,..., X_i,...X_N]$ as the population of CMPSOWV with population size of *N*. The multi-swarm approach is used to divide the population of CMPSOWV into *K* sub-swarms. Each *k*-th sub-swarm is denoted as $\mathbf{P}^{sub,k} = [X_1, ..., X_i, ...X_{Nk}]$ with a sub-swarm size of N^k, where $k = 1, ..., K$ and $N^k = \lfloor N/K \rfloor$.

The procedures of dividing the CMPSOVW population of $\mathbf{P} = [X_1,..., X_i,...X_N]$ into *K* sub-swarms are explained in Fig. 2. A reference point denoted as U^k is randomly generated to produce each of the *k*-th sub-swarm represented as $\mathbf{P}^{sub,k} = [X_1^k, ..., X_i^k, ...X_{N^k}^k]$. The *i*-th particle with personal best position of $P_{best,i}$ that has the nearest Euclidean distance to U^k is first identified and the remaining $N^k - 1$ particles with personal best positions that have the nearest Euclidean distances to$P_{best,i}$ are then combined to obtain the *k*-th sub-warm of $\mathbf{P}^{sub,k}$. All sub-swarm members of $\mathbf{P}^{sub,k}$ are then eliminated from the main population $\mathbf{P}$. Similar procedures are repeated until the main population $\mathbf{P}^{sub,k}$ is divided into *K* sub-swarms.

After obtaining all sub-swarms, two modified search operators are introduced into CMPSOWV to update the new position X_i^k of each *i*-th particle in the *k*-th sub-swarm based on different direction information. For the first sub-swarm where $k = 1$, define $P_{best,i}^k$ and G_{best} as the personal best position of the *i*-th particle and the global best particle, respectively. Let $P_{best,a}^k$ and $P_{best,b}^k$ be the personal best position of two randomly selected particles in the first sub-subswarm, where $a \neq b \neq i$ and $a, b \in [1, N^k]$. Then, the new position X_i^k of each *i*-th particle in the first sub-swarm is calculated as:

$$X_i^k = c_1 r_1 P_{best,i}^k + c_2 r_2 G_{best} + c_3 r_3 \left(P_{best,a}^k - P_{best,b}^k\right), \quad \text{for } k = 1 \tag{8}$$

where c_1, c_2 and c_3 are the acceleration coefficients; r_1, r_2 and r_3 are the random numbers generated from uniform distribution with values between 0 and 1. The first two components of Eq. (8) are same as those of Eq. (7), while the third component is added enhance the diversity of sub-swarm by allowing each particle to perform searching based on different direction information obtained from the randomly selected sub-swarm members.

For the remaining sub-swarms with indices of $k = 2, ..., K$, define $P^{SubBest,k-1}$ as the personal best position of the best particle in the previous (k - 1)-th sub-swarm. Let $P_{best,c}^k$ be the personal best position of a randomly selected particle in the *k*-th sub-swarm, where $c \neq i$ and$c \in [1, N^k]$. Then, the new position X_i^k of each *i*-th particle in these sub-swarms is updated as:

$$X_i^k = c_1 r_1 P_{best,i}^k + c_2 r_2 G_{best} + c_3 r_3 \left(P^{SubBest,k-1} - P_{best,c}^k\right), \quad \text{for } k = 2, ..., K \tag{9}$$

Similar with Eq. (8), the first two components of Eq. (9) are same as those of PSOWV. The third component of Eq. (9) is added to facilitate the information exchange between all particles in the *k*-th sub-swarm and the best particles in the (k - 1)-th sub-swarm in order to prevent the premature convergence of the *k*-th sub-swarm.

The pseudocode used to explain the current swarm evolution of CMPSOWV is presented in Fig. 3. The boundary checking is first

Algorithm 3: Current Swarm Evolution

Input: k, X_i^k, $P_{best,i}^k$, $f\left(P_{best,i}^k\right)$, $\varphi\left(P_{best,i}^k\right)$, G_{best}, $f\left(G_{best}\right)$, $\varphi\left(G_{best}\right)$

If $k = 1$ **then** /*Check if the particles belong to first sub-swarm*/
  For $i = 1$ to N^k **do**
    Randomly select $P_{best,a}^k$ and $P_{best,b}^k$ from the sub-swarm;
    Calculate the new position X_i^k of i-th particle using Eq. (8);
    Ensure the new position X_i^k is within search boundary;
    Evaluate the $f\left(X_i^k\right)$ and $\varphi\left(X_i^k\right)$ of new position X_i^k;
    Update the $P_{best,i}^k$, $f\left(P_{best,i}^k\right)$ and $\varphi\left(P_{best,i}^k\right)$ using Deb's rule **(Algorithm 1)**;
    Update the G_{best}, $f\left(G_{best}\right)$ and $\varphi\left(G_{best}\right)$ using Deb's rule **(Algorithm 1)**;
  End For
Else /*The particles belong to other sub-swarms*/
  For $i = 1$ to N^k **do**
    Determine $P^{SubBest,k-1}$ from the previous sub-swarm using Deb's rule **(Algorithm 1)**;
    Randomly select $P_{best,c}^k$ from the sub-swarm;
    Calculate the new position X_i^k of i-th particle using Eq. (9);
    Ensure the new position X_i^k is within search boundary;
    Evaluate the $f\left(X_i^k\right)$ and $\varphi\left(X_i^k\right)$ of new position X_i^k;
    Update the $P_{best,i}^k$, $f\left(P_{best,i}^k\right)$ and $\varphi\left(P_{best,i}^k\right)$ using Deb's rule **(Algorithm 1)**;
    Update the G_{best}, $f\left(G_{best}\right)$ and $\varphi\left(G_{best}\right)$ using Deb's rule **(Algorithm 1)**;
  End For
End If

Output: X_i^k, $f\left(X_i^k\right)$, $\varphi\left(X_i^k\right)$, $P_{best,i}^k$, $f\left(P_{best,i}^k\right)$, $\varphi\left(P_{best,i}^k\right)$, G_{best}, $f\left(G_{best}\right)$, $\varphi\left(G_{best}\right)$

Fig. 3. The pseudocode of current swarm evolution in the proposed CMPSOWV.

performed on the new position X_i^k obtained to ensure the upper and lower limits of all decision variables are satisfied. The fitness and total constraint violation of i-th particle in the k-th sub-swarm, denoted as $f(X_i^k)$ and $\varphi(X_i^k)$, respectively, are then evaluated. Based on these information, the Deb's rule as explained in the Fig. 1 are used to update the personal best position, personal best fitness and personal best value of total constraint violation for the i-th particle in the k-th sub-swarm denoted as $P_{best,i}^k$,$f(P_{best,i}^k)$ and $\varphi(P_{best,i}^k)$, respectively. Similar procedures are also used to update the global best particle, global best fitness and global best value of total constraint violation represented using G_{best}, $f(G_{best})$ and $\varphi(G_{best})$, respectively.

3.3. Memory swarm evolution

Substantial studies reported that an algorithm with single search operator can only perform well in certain types of problems due to the limited exploration and exploitation strengths of particles (Li et al., 2012; Lim et al., 2018; Vrugt et al., 2009; Wang et al., 2011). Therefore, an alternate evolution phase known as memory swarm evolution is incorporated into CMPSOWV to tackle different types of COPs more effectively by allowing each particle to perform searching with different exploration and exploitation strengths via the multiple search operator concept. Particularly, the memory swarm evolution of CMPSOWV is used to evolve the personal best position of each particle further by exploiting the useful information contained in other population members using two new search operators.

After completing the current swarm evolution, all sub-swarms denoted as $\mathbf{P}^{sub,k} = [X_1^k, ..., X_i^k, ...X_{Nk}^k]$ for $k = 1, ..., K$, obtained are recombined to form the main population of $\mathbf{P} = [X_1, ..., X_i, ...X_N]$ as follow:

$$\mathbf{P} = \mathbf{P}^{sub,1} \cup ... \cup \mathbf{P}^{sub,k} \cup ... \cup \mathbf{P}^{sub,K} \tag{10}$$

For each i-th particle, a random e-th particle is selected from the population and the personal best position of these two particles, denoted as $P_{best,i}$ and $P_{best,e}$, where $e \in [1, N]$ and $e \neq i$ are compared using Deb's rule.

If a randomly selected e-th particle is worse than the i-th particle, a search operator inspired by the differential evolution (Das & Suganthan, 2011; Das, Mullick & Suganthan, 2016) is employed to enhance the exploration search of CMPSOWV. Let $P_{best,f,d}$,$P_{best,g,d}$ and $P_{best,h,d}$ be the d-th component of the personal best positions of three particles randomly selected from population, where $f, g, h \in [1, N]$ and $f \neq g \neq h \neq i$. Suppose that $P_{best,i,d}^{new}$ refers to the d-th component of the new personal best position of i-th particle, then:

$$P_{best,i,d}^{new} = \begin{cases} P_{best,f,d} + \psi\left(P_{best,g,d} - P_{best,h,d}\right), & \text{if } r_a > 0.5 \\ P_{best,i,d}, & \text{otherwise} \end{cases}, \quad \text{if } P_{best,e} \text{ is worse than } P_{best,i} \tag{11}$$

where r_a is a random number generated from the uniform distribution with values between 0 and 1, i.e., $r_a \in [0, 1]$; $\psi$is a random number generated from the uniform distribution with value between -1 and 1, i.e., $\psi \in [-1, 1]$.

In contrary, a search operator inspired by the teaching-learning-based optimization (Zou, Chen & Xu, 2019) is employed to promote the exploitation search of CMPSOWV if the randomly selected e-th particle is better than the i-th particle. Let$P_{best,l}$and$P_{best,m}$be the personal best positions of two randomly selected particles to maintain the population diversity, where $l, m \in [1, N]$ and$l \neq m \neq i$. The new personal best position of each i-th particle denoted as $P_{best,i}^{new}$ is

Algorithm 4: Memory Swarm Evolution

Input: $P_{best,i}, f(P_{best,i}), \varphi(P_{best,i}), G_{best}, f(G_{best}), \varphi(G_{best})$

01: Randomly select the e-th particle with personal best position $P_{best,e}$;
02: Compare $P_{best,i}$ and $P_{best,e}$ using Deb's rule **(Algorithm 1)**;
03: **If** $P_{best,e}$ is worse than $P_{best,i}$ **then**
04: **For** $d = 1$ to D **do**
05: Calculate $P^{new}_{best,i,d}$ using Eq. (11);
06: **End For**
07: **Else If** $P_{best,e}$ is better than $P_{best,i}$ **then**
08: Calculate $P^{new}_{best,i}$ using Eq. (12);
09: **End If**
10: Ensure the new personal best position $P^{new}_{best,i}$ is within search boundary;
11: Evaluate the $f(P^{new}_{best,i})$ and $\varphi(P^{new}_{best,i})$ of new personal best position $P^{new}_{best,i}$;
12: Update the $P_{best,i}$, $f(P_{best,i})$ and $\varphi(P_{best,i})$ using Deb's rule **(Algorithm 1)**;
13: Update the G_{best}, $f(G_{best})$ and $\varphi(G_{best})$ using Deb's rule **(Algorithm 1)**;

Output: $P_{best,i}, f(P_{best,i}), \varphi(P_{best,i}), G_{best}, f(G_{best}), \varphi(G_{best})$

Fig. 4. The pseudocode of memory swarm evolution in the proposed CMPSOWV.

then calculated as:

$$P^{new}_{best,i} = P_{best,i} + r_b(P_{best,e} - P_{best,i}) + r_c(P_{best,l} - P_{best,m}), \quad \text{if } P_{best,e} \text{ is better than } P_{best,i} \tag{12}$$

where r_b and r_c are two random number generated from a uniform distribution with values between 0 and 1, i.e., $r_b, r_c \in [0, 1]$.

The pseudocode for the memory swarm evolution of CMPSOWV is presented in Fig. 4. Boundary checking is used to ensure the new personal best position $P^{new}_{best,i}$ obtained can satisfy the upper and lower limits of all decision variables. The fitness and total constraint violation associated with $P^{new}_{best,i}$, denoted as $f(P^{new}_{best,i})$ and $\varphi(P^{new}_{best,i})$, respectively, are then evaluated. Deb's rule are applied to determine if the $P^{new}_{best,i}$ of i-th particle can be used to replace its original personal best position $P_{best,i}$ and the global best particle G_{best}.

3.4. Mutation

Global best particle is considered as the most influential source of PSO variants in guiding the population to search towards the promising regions of search space. If the global best particle is trapped into local optima, the remaining particles tend to converge towards the inferior regions of search space and this leads to premature convergence. Therefore, a probabilistic mutation operator is incorporated into CMPSOWV to provide an additional momentum for the global best particle in escaping from the local optima regions.

After completing both evolution phases of CMPSOWV in every generation, mutation is performed on the global best particle with a mutation probability of $P^{MUT} = 1/D$, where D refers to total dimension size. Let $d_r \in [1, D]$be a random dimension chosen from the global best particle of G_{best} for mutation. Suppose that X^L_d and X^U_drepresent the lower and upper boundary limits of decision variable in any d-th dimension, respectively. Then, each of the d-th dimension of new global best particle denoted as $G^{new}_{best,d}$ can be obtained as:

$$G^{new}_{best,d} = \begin{cases} G_{best,d} + r_d(X^U_d - X^L_d), & \text{if } d = d_r \\ G_{best,d}, & \text{otherwise} \end{cases} \tag{13}$$

where r_d is a random number generated from a uniform distribution with values between -1 and 1, i.e., $r_d \in [-1, 1]$.

The pseudocode for the mutation operation of CMPSOWV is summarized in Fig. 5. Boundary checking is used to ensure the new global best particle G^{new}_{best} obtained can satisfy the upper and lower limits of all decision variables. The fitness and total constraint violation values associated with G^{new}_{best}, denoted as $f(G^{new}_{best})$ and $\varphi(G^{new}_{best})$, respectively, are then evaluated. Finally, Deb's rule is applied to determine if the new global best particle G^{new}_{best} can be used to replace the original global best particle G_{best}.

3.5. Overall framework of the proposed CMPSOWV

The complete framework of CMPSOWV is presented in Fig. 6. The initial population of CMPSOWV is randomly initialized, followed by the evaluation of fitness and total constraint violation associated with each particle. The current swarm evolution and memory swarm evolution are performed to update the current position and personal best position of each particle, respectively. A probabilistic mutation operator is applied on the global best particle to prevent the swarm stagnation in the local optima regions of search space. These processes are repeated until the termination criterion of algorithm is satisfied. In this research, letγbe the fitness evaluation counter that is updated whenever the fitness and total constraint violation of a particle is evaluated. The CMPSOWV is then terminated ifγexceeds the maximum fitness evaluation number represented as$\Upsilon_{\max}$.

4. Simulation results

4.1. Benchmark functions

The performance of the proposed CMPSOWV are evaluated using 22 selected benchmark functions of CEC 2006 (Liang et al., 2006). Although there were 24 functions introduced in CEC 2006, previous studies reported that almost none of the existing algorithms are able to locate the feasible regions of functions G20 and G22 due to the presence of large number of active constraints (Elsayed et al., 2014). Therefore, these two functions are not used for the performance comparison. Define D as the number of decision variable and $\rho = |\Omega|/|S|$ as the estimated ratio between the feasible region of$\Omega$and overall search space of S. Let the numbers of linear equality constraints, nonlinear inequality constraints, linear equality constraints and nonlinear equality constraints of each

Algorithm 5: Mutation On Global Best Particle

Input: $G_{best}, f(G_{best}), \varphi(G_{best}), D, X^U, X^L$

Randomly select the d_r-th dimension of global best particle for mutation;
For $d = 1$ to D **do**
    Calculate $G^{new}_{best,d}$ using Eq. (13);
End For
Ensure the new personal best position G^{new}_{best} is within search boundary;
Evaluate the $f(G^{new}_{best})$ and $\varphi(G^{new}_{best})$ of new global best particle G^{new}_{best};
Update the $G_{best}, f(G_{best})$ and $\varphi(G_{best})$ using Deb's rule **(Algorithm 1)**;

Output: $G_{best}, f(G_{best}), \varphi(G_{best})$

Fig. 5. The pseudocode of probabilistic mutation operator in the proposed CMPSOWV.

Main Algorithm: CMPSOWV

Input: $N, K, D, X^U, X^L, \Upsilon_{\max}$

Randomly generate the initial population of CMPSOWV and initialize $\gamma = 0$;
While $\gamma < \Upsilon_{\max}$ **do**
    Formation of Multi-Swarm (**Algorithm 2**); */*Divide the population into K sub-swarms*/*
    For $k = 1$ to K **do** */*Perform current swarm evolution in each sub-swarm*/*
        Current Swarm Evolution (**Algorithm 3**);
        $\gamma \leftarrow \gamma + N^k$; */*Update fitness evaluation counter*/*
    End for
    Formation of main population using Eq. (10);
    For $i = 1$ to N **do** */*Perform memory swarm evolution in main populations*/*
        Memory Swarm Evolution (**Algorithm 4**);
        $\gamma \leftarrow \gamma + 1$; */*Update fitness evaluation counter*/*
    End For
    Randomly generate a number of $rand \in [0,1]$;
    If $rand < P^{MUT}$ **then**
        Mutation On Global Best Particle (**Algorithm 5**);
        $\gamma \leftarrow \gamma + 1$; */*Update fitness evaluation counter*/*
    End If
End While

Output: $G_{best}, f(G_{best}), \varphi(G_{best})$

Fig. 6. The complete framework of CMPSOWV.

tested function to be represented as *#LI, #NI, #LE* and *#NE*, respectively. Suppose that *#A* is the number of active constraints at the global optimum, while $f(X^*)$ refers to the objective function value associated with global optimumX^*. Then, the characteristics of 22 selected benchmark functions in CEC 2006 are summarized in Table 1.

4.2. Algorithms comparisons and parameter settings

The optimization performance of the proposed CMPSOWV in solving the 22 selected CEC 2006 benchmark functions are compared with six constrained optimization algorithm knowns as: constrained PSO with a shake mechanism (CPSO-Shake) (Cagnina et al., 2011), improved PSO (IPSO) (Mezura-Montes & Flores-Mendoza, 2009), constraint violation with interval arithmetic PSO (CVI-PSO) (Mazhoud et al., 2013), elitist teaching-learning-based optimization (ETLBO) (Rao & Patel, 2012), artificial bee colony (ABC) (Karaboga & Basturk, 2007) and modified artificial bee colony (MABC) (Mezura-Montes & Cetina-Domínguez, 2012). These six algorithms are selected for performance comparisons due to their excellent performance in solving COPs. The parameter settings for all compared algorithms are set based on the recommendation of their original literature and presented in Table 2. Similar with the proposed CMPSOWV, Deb's method was employed by CPSO-Shake, ETLBO, ABC and MABC as the constraint handling technique. A modified Deb's method was introduced into IPSO where the violation of sum for equality and inequality constraints were handled separately and compared using dominance criterion in order to provide more detailed information in selection process. For CVI-PSO, the total constraint violations were evaluated and normalized using a feasibility-based rule developed using the interval arithmetic method, aiming to attract the problem solutions to the feasible regions of search space.

The parameter settings of CMPSOWV are explained as follows. The population size is set as $N = 100$ with the subpopulation size of$N^k = 10$for$k = 1, ..., 10$. Meanwhile, the acceleration coefficients of CMPSOWV are set as the equal values of$c_1 = c_2 = c_3 = 4.1/3$. Detailed procedures used to determine the appropriate values of these parameter settings will be explained in the following subsections. The tolerance of equality constraint is set as $\delta = 0.0001$according to Liang et al. (2006). Similar with most compared methods, the maximum fitness evaluation numbers of CMPSOWV to solve all tested functions are set as$\Upsilon_{\max} = 240,000$. Each benchmark function is simulated with 25 independent runs using Matlab 2017a on the personal computer with Intel ® Core i7-7500 CPU @ 2.70 GHz.

Table 1
Characteristics of 22 selected benchmark functions in CEC 2006.

No.	D	Type of Function	$\rho(\%)$	$\#LI$	$\#NI$	$\#LE$	$\#NE$	$\#A$	$f(X^*)$
G01	13	Quadratic	0.0111	9	0	0	0	6	−15.0000000000
G02	20	Nonlinear	99.9971	0	2	0	0	1	− 0.8036191042
G03	10	Polynomial	0.0000	0	0	0	1	1	−1.0005001000
G04	5	Quadratic	51.1230	0	6	0	0	2	−30,665.5386717834
G05	4	Cubic	0.0000	2	0	0	3	3	5126.4967140071
G06	2	Cubic	0.0066	0	2	0	0	2	−6961.8138755802
G07	10	Quadratic	0.0003	3	5	0	0	6	24.3062090681
G08	2	Nonlinear	0.8560	0	2	0	0	0	−0.0958250415
G09	7	Polynomial	0.5121	0	4	0	0	2	680.6300573745
G10	8	Linear	0.0010	3	3	0	0	0	7049.2480205286
G11	2	Quadratic	0.0000	0	0	0	1	1	0.7499000000
G12	3	Quadratic	4.7713	0	1	0	0	0	−1.0000000000
G13	5	Nonlinear	0.0000	0	0	0	3	3	0.0539415140
G14	10	Nonlinear	0.0000	0	0	3	0	3	−47.7648884595
G15	3	Quadratic	0.0000	0	0	1	1	2	961.7150222899
G16	5	Nonlinear	0.0204	4	34	0	0	4	−1.9051552586
G17	6	Nonlinear	0.0000	0	0	0	4	4	8853.5396748064
G18	9	Quadratic	0.0000	0	13	0	0	6	−0.8660254038
G19	15	Nonlinear	33.4761	0	5	0	0	0	32.6555929502
G21	7	Linear	0.0000	0	2	3	1	6	193.7245100700
G23	9	Linear	0.0000	0	2	3	1	6	−400.0551000000
G24	2	Linear	79.6556	0	2	0	0	2	−5.5080132716

Table 2
Parameter settings of all compared algorithms in solving CEC 2006 benchmark functions.

Algorithm	Parameter Settings
CMPSOWV	Population size $N = 100$, subpopulation size$N^k = 10$ with $k = 1,...,10$, acceleration coefficients $c_1 = c_2 = c_3 = 4.1/3$
CPSO-Shake	$N = 10$, c_1,c_2,$c_3 \in [1.4, 1.9]$, constriction factor$\chi \in [0.4, 0.9]$, maximum mutation rate $p_m^{\max} = 0.4$, minimum mutation rate $p_m^{\min} = 0.1$.
IPSO	$N = 80$,$\chi = 0.729$, $c_1 = 0$(self-cognitive term is eliminated),$c_2 = 2.5$, probability$p \in [0.62, 0.82]$
CVI-PSO	$N = 50$, $c_1 = c_2 = 2.0$, inertia weight ω: $1.0 - 0.5$
ETLBO	$N = 100$, elite size, $\varepsilon = 4$, teaching factor$T_f \in \{1, 2\}$
ABC	$N = 40$, modification rate $MR = 0.8$, limit $= 0.5 \times sn \times D$, scout production period$SPP = 0.5 \times sn \times D$
MABC	$N = 20$, $MR = 0.8$, limit $= 145$, decreasing rate$dec = 1.002$

4.3. Performance metrics

The performances of all compared algorithms are measured using the mean fitness value and standard deviation denoted as F_{mean} and SD, respectively. Particularly, the F_{mean} value indicates the searching accuracy of algorithm by comparing the fitness solution produced and the actual global optimum, while the SD value is used to evaluate the consistency of an algorithm to solve a benchmark problem with similar performance.

A set of non-parametric statistical procedures are used for rigorous performance comparison between CMPSOWV and its peers (Derrac, García, Molina & Herrera, 2011; García, Molina, Lozano & Herrera, 2008). Wilcoxon signed rank test (García et al., 2008) is first used to perform pairwise comparison between CMPSOWV and its peers at the significance level of 5%, i.e., $\alpha = 0.05$, and the results are presented as the R^+, R^-, p, and h values. R^+ and R^- indicate the sum of ranks where CMPSOWV outperforms and underperforms the compared peers, respectively. The p-value is the minimum level of significance to detect performance differences between algorithms. If the p-value obtained is smaller thanα, it implies that the better results achieved by the best algorithm is statistically significant. Referring to the p-value andα, the h value can be used to indicate if CMPSOWV is significantly better (i.e., $h =$ "+"), insignificant (i.e., $h =$ "=") or significantly worse (i.e., $h =$ "-") than the compared algorithms at the statistical point of view.

For the non-parametric statistical analysis that involve multiple comparison of algorithms, the Friedman test is first conducted to obtain the average ranking of each compared methods and determine the global differences between these algorithms via the obtained p-values (Derrac et al., 2011; García et al., 2008). If the significant global different is identified, three post-hoc procedures (Derrac et al., 2011) known as the Bonferroni-Dunn, Holm and Hochberg are then used to characterize the concrete differences among all compared methods based on the adjusted p-values obtained.

4.4. Parameter sensitivity analysis

As explained in Section 3, a multi-swarm approach is introduced into the current swarm evolution of CMPSOWV, aiming to preserve the diversity of population. It is anticipated that some parameters such as the population size, subpopulation size and acceleration coefficients can play crucial roles in governing the search behaviour of CMPSOWV. In this section, a series of parameter sensitivity analyses to investigate the impacts of these parameter settings on the searching performance of CMPSOWV and the proper setting of these parameters.

4.4.1. Effects of population sizes

The effects of population sizes on the search performance of CMPSOWV are investigated by setting $N = 20$, 60, 100, 140 and 180 using ten selected CEC 2006 benchmark functions with different characteristics, i.e., functions G01, G02, G05, G07, G10, G13, G14, G17, G19 and G23. Each selected N value is simulated 25 times with the subpopulation size of $N^k = 10$for$k = 1$, ..., 10 and maximum fitness evaluation number of$\Upsilon_{\max} = 240,000$. The F_{mean} values obtained by the proposed CMPSOWV in solving these ten selected CEC 2006 benchmark functions using different values of N are summarized in Fig. 7.

Generally, the search accuracy of CMPSOWV tends to be deteriorated when smaller population sizes are set. It is reported that

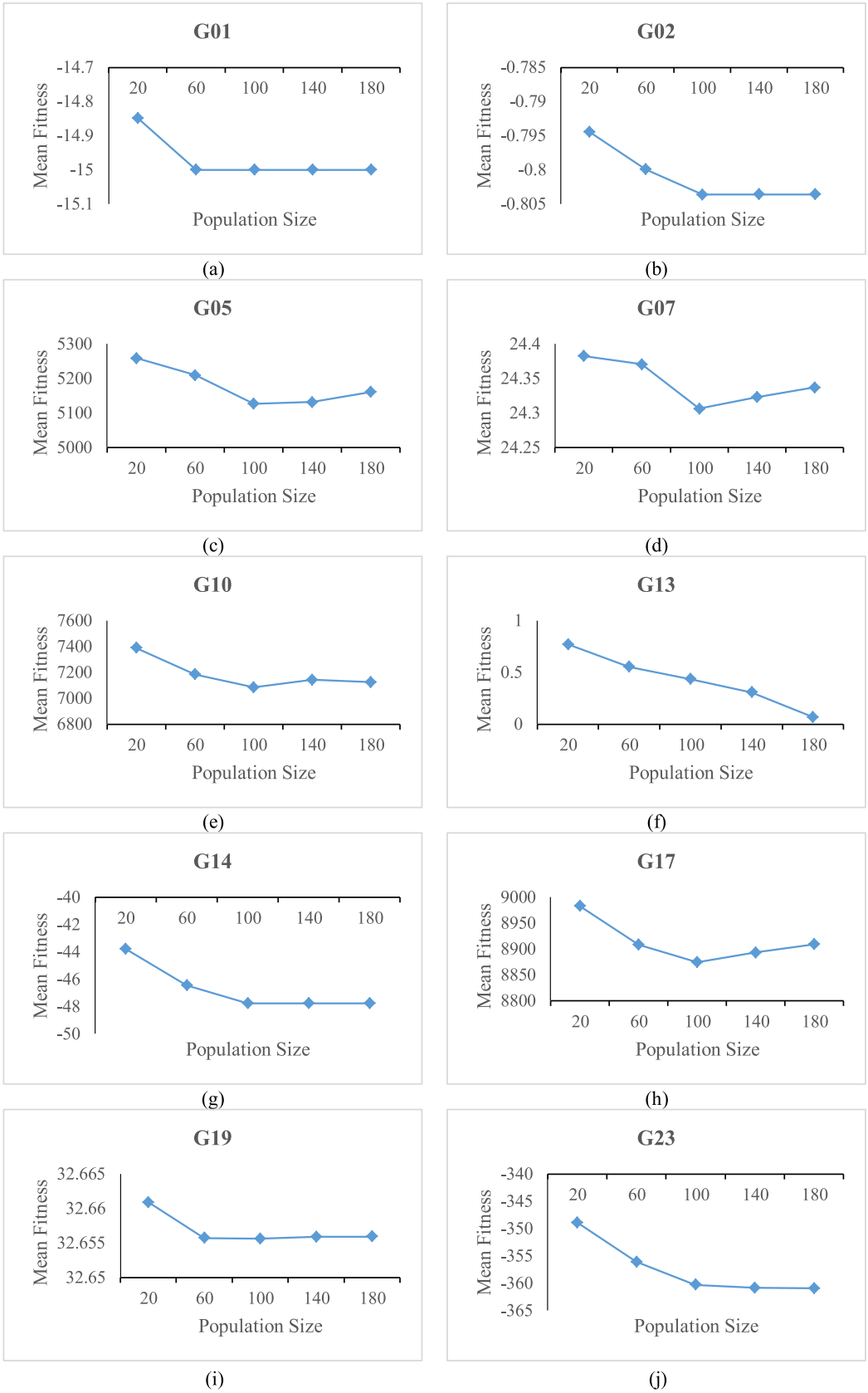

Fig. 7. The F_{mean} values produced by CMPSOWV for functions (a) G01, (b) G02, (c) G05, (d) G07, (e) G10, (f) G13, (g) G14, (h) G17, (i) G19 and (j) G23 with different population sizes.

CMPSOWV with the population size of $N=20$ consistently produces the worst F_{mean} values for all for ten selected CEC 2006 benchmark functions. Given the fixed number of subpopulation size, the lower population size produces smaller number of subswarms for the current swarm evolution, hence introducing lower diversity in population that might lead to the premature convergence issue. Meanwhile, the similar search performances are demonstrated by CMPSOWV in solving the functions G01, G02, G14, G19 and G23 when larger population sizes of $N=100$, 140 and 180 are set. For other tested functions such as G05, G07, G10 and G17, it is observed that the search accuracy of CMPSOVW tends to be degraded when larger population sizes such as $N=140$ and 180 are set. The excessive diversity brought by the larger numbers of sub-swarms tend to prevent the algorithm converging towards the promising regions of search space, hence producing the inferior F_{mean} values. Based on the simulation results summarized in Fig. 7, the population size of $N=100$ is reported to be an appropriate parameter setting for the following performance evaluations because the proposed CMPSOWV is able to solve majority of the tested function with the best F_{mean} values.

4.4.2. Effects of subpopulation sizes

The impacts of subpopulation sizes in influencing the search accuracy of CMPSOWV are further investigated by setting$N^k=$ 4, 5, 10, 20 and 25 to solve the same ten selected CEC 2006 benchmark functions mentioned earlier. Similarly, each of theN^kvalue considered in this parameter sensitivity analysis is run 25 times with the population size of $N=100$ and maximum fitness evaluation number of$\Upsilon_{\max}=240,\ 000$. Fig. 8 illustrates the F_{mean} values obtained by the proposed CMPSOWV in solving all selected CEC 2006 benchmark functions using different values ofN^k.

From Fig. 8, it is observed that the search performances of CMPSOWV in tackling certain benchmark problems are compromised when the subpopulation sizes used are too small. For instance, CMPSOWV produces the relatively inferior values of F_{mean} when it is used to solve the functions G01, G02, G05, G10, G13 and G23 with the subpopulation sizes of $N^k=$ 4 and 5. Given the same population size, larger numbers of sub-swarms are produced during the current swarm evolution of CMPSOWV when the subpopulation size is set too low. The excessive diversity induced by these large number of sub-swarms tends to misguide the particle searching towards the other unexplored regions of search space before it manages to locate the sufficiently promising regions, hence explaining the unsatisfactory search accuracy of algorithm. On the other hand, the relatively inferior simulation results obtained by CMPSOWV to solve the functions G07, G14, G17 and G19 indicate the drawback of setting too large values for the subpopulation sizes (i.e., $N^k=$ 20 and 25). Under these circumstances, the number of sub-swarms produced in current swarm evolution of CMPSOWV are smaller and the limited information exchanges among these subswarms might exacerbated the ability of algorithm to search towards the optimal region of search space. Finally, it is observed that the search performances of CMPSOWV in solving all ten selected benchmark functions are compelling when the subpopulation size is set as$N^k=$ 10 by producing the best F_{mean} values for the functions G01, G02, G05, G07, G10 G13, G14, G17 and G19. Based on the simulation results obtained from this parameter sensitivity analysis, the value of$N^k=$ 10 is chosen as the suitable subpopulation sizes of CMPSOWV for the following performance evaluations.

4.4.3. Effects of acceleration coefficients

The proper settings of acceleration coefficients for CMPSOVW is another main concern of current study because these parameters have crucial impact in governing the dynamic behaviour of particles. Let c be the sum of acceleration coefficients of CMPSOWV, where $c=c_1+c_2+c_3$. Extensive theoretical studies were conducted earlier to investigate the effects of acceleration coefficients on the particle's trajectory and most studies have recommended to set the total acceleration coefficients as 4.1 to ensure the convergence of PSO (Clerc & Kennedy, 2002; van den Bergh & Engelbrecht, 2006). Similarly, the empirical results obtained in this study also suggest the condition of $c=4.1$ to be satisfied in order for CMPSOWV achieve the convergence behaviour.

Although the total acceleration coefficients of CMPSOWV is set as $c=4.1$ based on the abovementioned literature studies and the empirical analysis conducted, it remains nontrivial to determine the proper values for each individual acceleration coefficient of c_1, c_2 and c_3. While there are infinite combinations of c_1, c_2 and c_3 to satisfy the condition of $c=4.1$, it is not realistic to evaluate the impact of every possible combination on the search performance of CMPSOVW. Four main configurations of c_1, c_2 and c_3 with $c=4.1$ are considered in the parameter sensitivity analysis instead (Xu & Xin, 2005), where: (a) test case #1 refers to the ratios ofc_1: c_2: $c_3=1$: 1: 2, (b) test case #2 refers to the ratios ofc_1: c_2: $c_3=1$: 2: 1, (c) test case #3 refers to the ratios ofc_1: c_2: $c_3=2$: 1: 1and (d) test case #4 refers to the ratios ofc_1: c_2: $c_3=1$: 1: 1. The same ten selected CEC 2006 benchmark functions are used to evaluate the search accuracy of CMPSOWV. Each test case considered in the parameter sensitivity analysis is run 25 times with the population size of $N=100$, subpopulation size of $N^k=$ 10 for $k=1,\ldots,10$, and maximum fitness evaluation number of $\Upsilon_{\max}=240,\ 000$ The F_{mean} values obtained by CMPSOWV in solving all selected CEC 2006 benchmark functions using the four test cases with different ratios of c_1: c_2: $c_3$are presented in Fig. 9.

Generally, the search performances of CMPSOWV in tackling the ten selected CEC 2006 benchmark functions are relatively poor when test case #1 is considered. The F_{mean} values obtained by CMPSOWV with the ratios of c_1: c_2: $c_3=1$: 1: 2are the most inferior in solving the functions G01, G02, G05, G07, G10, G13, G14 and G23. The higher weightage of acceleration coefficient on the third component of Eqs. (8) and (9) tends to introduce excessive diversity on the CMPSOWV population and prevent it from locating the promising regions of search space during the optimization process. Meanwhile, it is reported that the proposed CMPSOWV with acceleration coefficients set under the test cases #2 and #3 can perform well in different benchmark functions. For instance, the CMPSOWV with the ratio of c_1: c_2: $c_3=1$: 2: 1(i.e., test case #2) has better search accuracy in solving functions G02, G05, G10, G13, G17 and G23 than that of withc_1: c_2: $c_3=2$: 1: 1(i.e., test case #3). On the other hand, CMPSOWV withc_1: c_2: $c_3=2$: 1: 1outperforms that withc_1: c_2: $c_3=1$: 2: 1in solving the benchmark functions of G01, G07, G14 and G19. Among all test cases considered, the proposed CMPSOWV with c_1: c_2: $c_3=1$: 1: 1(i.e., test case #4) has demonstrated the most promising search accuracy because it produces eight best F_{mean} values in solving the ten selected CEC 2006 benchmark functions, i.e., functions G01, G02, G05, G07, G10, G14, G17 and G23. Based on these findings, all acceleration coefficients of CMPSOWV are set as the equal constants of$c_1=c_2=c_3=4.1/3$because these parameter settings are able to achieve better balancing of exploration and exploitation searches of algorithm as compared to other test cases.

4.5. Performance evaluation using CEC 2006 benchmark functions

4.5.1. Comparison of f_{mean} and SD results

The simulation results of mean fitness (F_{mean}) and standard deviation (SD) obtained by the CMPSOWV and six selected competitors in solving all benchmark problems are presented in Table 3. The best results are indicated in boldface, while "-" implies that the results are not available because the F_{mean} and SD values of the six competitors are extracted from their corresponding literatures. The performance comparison between CMPSOWV and its

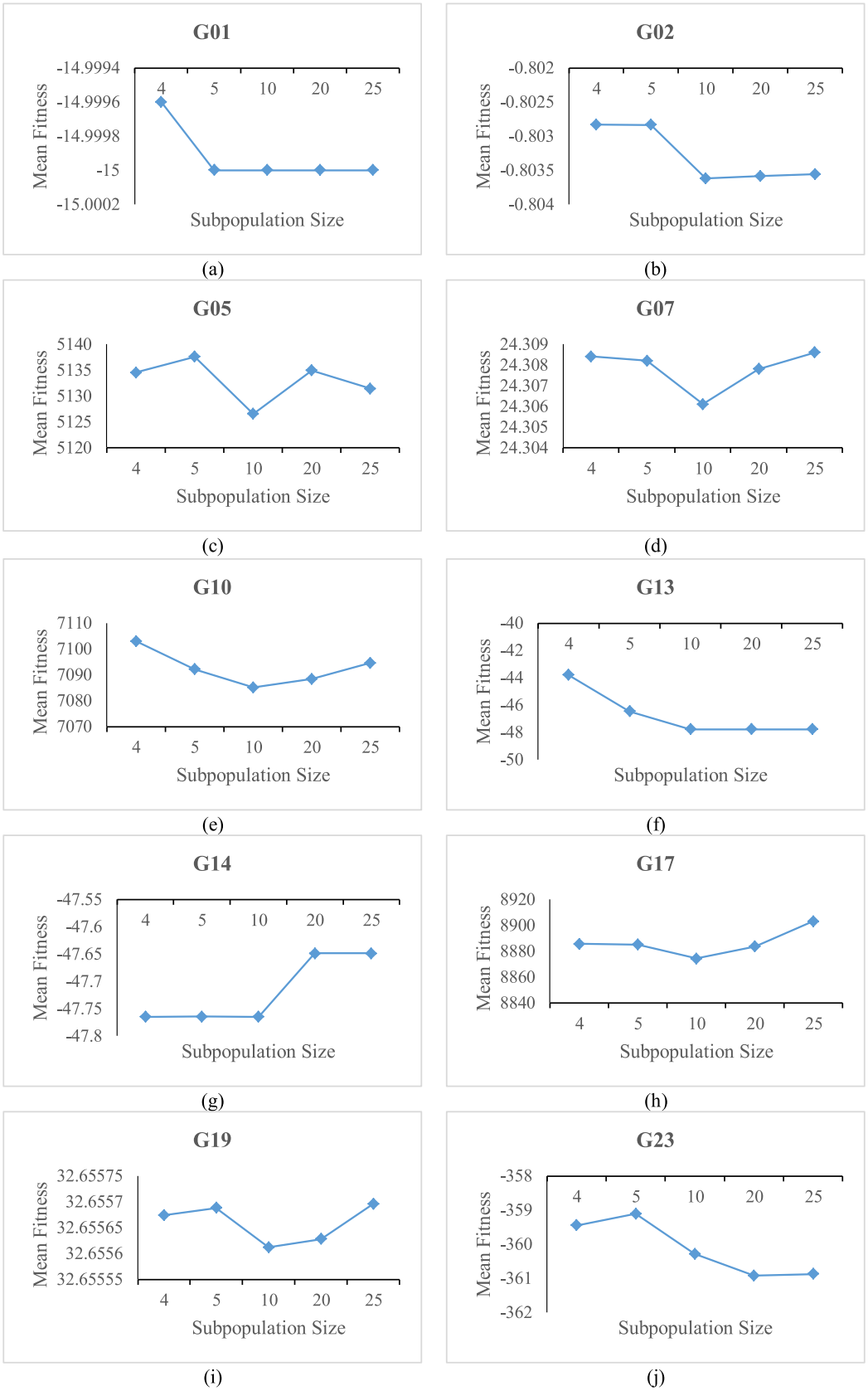

Fig. 8. The F_{mean} values produced by CMPSOWV for functions (a) G01, (b) G02, (c) G05, (d) G07, (e) G10, (f) G13, (g) G14, (h) G17, (i) G19 and (j) G23 with different subpopulation sizes.

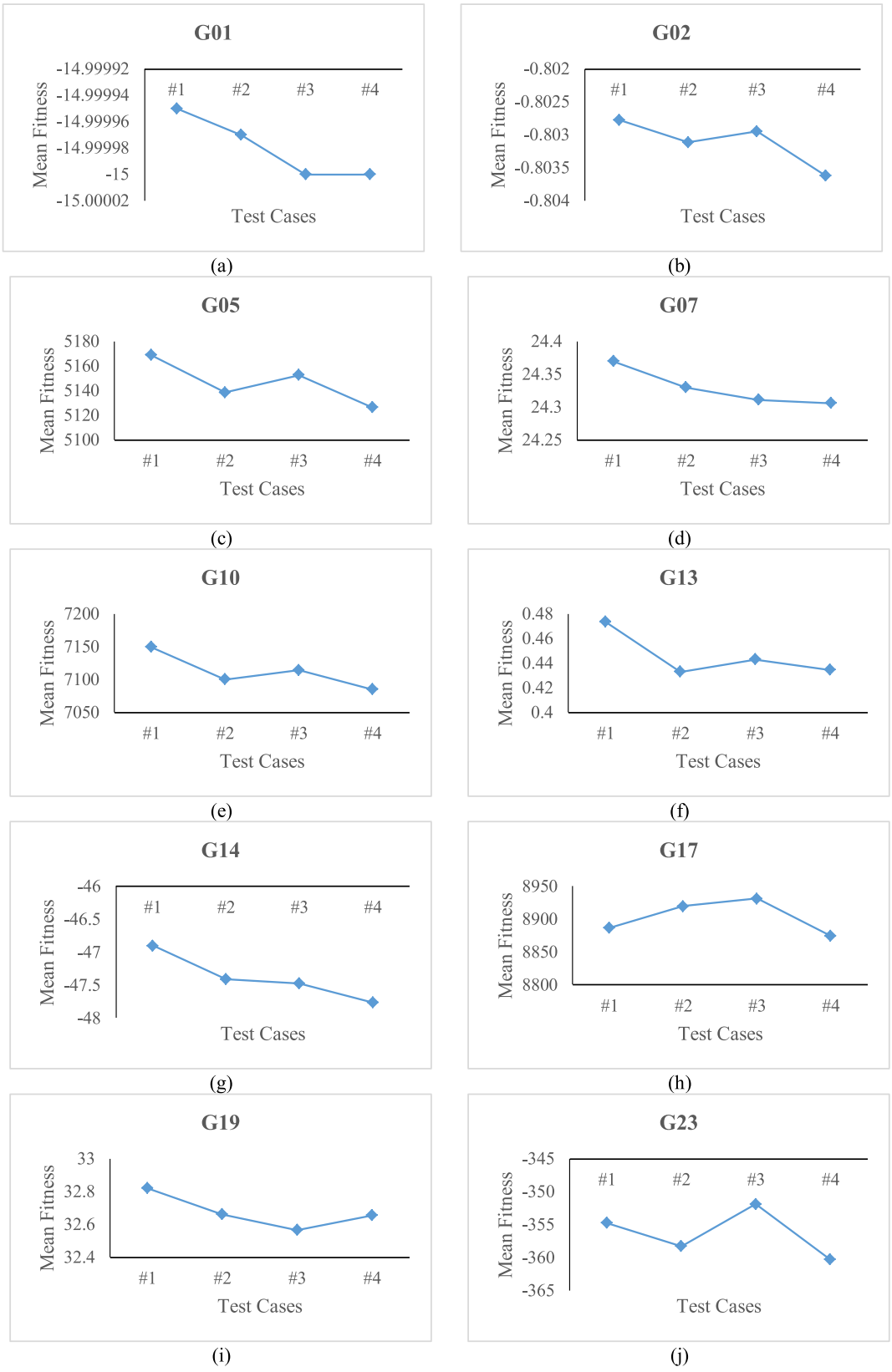

Fig. 9. The F_{mean} values produced by CMPSOWV for functions (a) G01, (b) G02, (c) G05, (d) G07, (e) G10, (f) G13, (g) G14, (h) G17, (i) G19 and (j) G23 with four different ratios of acceleration coefficients as specified in all test cases considered, where test case #1 refers to c_1: c_2: c_3 = 1: 1: 2, test case #2 refers to c_1: c_2: c_3 = 1: 2: 1, test case #3 refers to c_1: c_2: c_3 = 2: 1: 1and test case #4 refers to c_1: c_2: c_3 = 1: 1: 1.

Table 3
Performance comparison between CMPSOWV with Six Peer algorithms in CEC 2006 Benchmark functions.

Function	Criteria	CMPSOWV	CPSO-Shake	IPSO	CVI-PSO	ETLBO	ABC	MABC
G01	F_{mean}	**−15.0000**	**−15.0000**	**−15.0000**	**−15.0000**	**−15.0000**	**−15.0000**	**−15.0000**
	SD	0.00E+00	–	0.00E+00	4.50E-15	0.00E+00	0.00E+00	0.00E+00
G02	F_{mean}	**−0.803619**	−0.79661	−0.713879	−0.790875	**−0.803619**	−0.792412	−0.799336
	SD	2.79E-07	–	4.62E-02	1.09E-02	0.00E+00	1.20E−02	6.84E−03
G03	F_{mean}	**−1.000500**	−1.**0005**	−0.1540	−0.9999	−1.0003	−1.000000	−1.000000
	SD	2.13E−11	–	1.70E-01	3.70E-16	1.40E−04	0.00E+00	4.68E−05
G04	F_{mean}	**−30,665.5390**	−30,646.179	**−30,665.5390**	**−30,665.5390**	**−30,665.5390**	**−30,665.539**	**−30,665.539**
	SD	0.00E+00	–	7.40E-12	1.00E-11	0.00E+00	0.00E+00	2.22E−11
G05	F_{mean}	**5126.4975**	5240.49671	5135.521	5127.2777	5168.7194	5185.7140	5178.1390
	SD	1.32E-01	–	1.23E+01	0.00E+00	5.41E+01	7.54E+01	5.61E + 01
G06	F_{mean}	**−6961.8140**	−6859.0759	**−6961.8140**	**−6961.8140**	**−6961.8140**	−6961.8130	**−6961.8140**
	SD	0.00E+00	–	2.81E-05	0.00E+00	0.00E+00	2.00E−03	0.00E+00
G07	F_{mean}	**24.3061**	24.912209	24.691	26.5612	24.3100	24.4730	24.4150
	SD	2.51E−11	–	2.20E-01	1.64E+00	7.11E−03	1.86E−01	1.24E−01
G08	F_{mean}	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**
	SD	0.00E+00	–	4.23E-17	0.00E+00	0.00E+00	0.00E+00	4.23E−17
G09	F_{mean}	**680.6300**	681.3730	680.6740	680.7557	**680.6300**	680.6400	680.6500
	SD	2.76E−12	–	3.00E-02	7.92E-02	0.00E+00	4.00E−03	1.55E−02
G10	F_{mean}	7085.1718	7850.4010	7306.4660	**7053.2143**	7143.4500	7224.4070	7233.8820
	SD	7.13E+02	–	2.22E+02	1.06E+01	1.13E+02	1.34E+02	1.10E+02
G11	F_{mean}	**0.749900**	**0.7499**	0.7530	**0.750000**	**0.749980**	**0.750000**	**0.750000**
	SD	1.03E−12	–	6.53E-03	0.00E+00	7.06E−05	0.00E+00	2.30E−05
G12	F_{mean}	**−1.0000**	**−1.0000**	**−1.0000**	**−1.0000**	**−1.0000**	**−1.0000**	**−1.0000**
	SD	0.00E+00	–	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
G13	F_{mean}	0.434464	0.450941	0.430408	**0.065590**	0.838510	0.968000	0.158552
	SD	1.59E−01	–	2.30E+00	1.02E-02	2.26E−01	5.50E−02	1.72E−01
G14	F_{mean}	**−47.76488**	−45.66588	−44.572	−44.42469	−43.805	−40.1071	−47.271
	SD	1.46E−07	–	1.58E+00	1.41E+00	2.32E+00	7.14E+00	2.46E−01
G15	F_{mean}	**961.71502**	962.51602	962.242	961.71859	962.044	966.2868	961.719
	SD	4.64E−13	–	6.20E-01	6.87E-04	4.39E−01	−3.12E+00	1.42E−02
G16	F_{mean}	**−1.905155**	−1.7951553	**−1.9052**	**−1.905155**	**−1.905155**	**−1.9052**	**−1.905**
	SD	3.48E−14	–	2.42E-12	8.52E-15	0.00E+00	2.34E-16	4.52E−16
G17	F_{mean}	8874.2046	8894.7087	8911.7380	**8853.5399**	8895.7544	8941.9245	8987.459
	SD	3.64E−01	–	2.73E+01	3.70E-12	5.14E+01	4.26E+01	9.57E+01
G18	F_{mean}	**−0.866025**	−0.7870254	−0.862842	−0.809109	−0.865755	−0.86587	−0.795019
	SD	3.64E−15	–	4.41E-03	6.27EE-02	5.09E−04	3.37E-04	9.39E−02
G19	F_{mean}	**32.655612**	64.505593	37.927	35.0673379	33.3699	36.0078	34.267
	SD	1.53E−05	–	3.20E+00	2.29E+00	7.87E−02	1.83E+00	6.31E−01
G21	F_{mean}	**193.75354**	193.77137	217.356	193.78693	206.118	275.5436	306.609
	SD	2.52E-03	–	2.65E+01	3.38E-05	2.99E+01	6.05E+01	1.98E+01
G23	F_{mean}	−360.2836	−271.8431	−99.598	**−400.0000**	−352.263	−4.3254	−35.272
	SD	1.43E+01	–	1.20E+02	0.00E+00	2.33E+01	1.37E+01	8.28E+01
G24	F_{mean}	**−5.508013**	**−5.508013**	**−5.508013**	**−5.508013**	**−5.508013**	**−5.508**	**−5.508013**
	SD	0.00E+00	–	0.00E+00	0.00E+00	0.00E+00	9.36E-16	2.71E−15
	w/t/l	–	16/6/0	13/7/2	10/8/4	12/10/0	15/7/0	13/8/1
	#BMF	**18**	6	7	12	10	7	8

six peers are also summarized as *w/t/l* and *#BMF*. Specifically, *w/t/l* implies that CMPSOWV outperforms the selected competitor in *w* functions, ties in *t* functions and loses in *l* functions. Meanwhile, *#BMF* represents the number of best F_{mean} values produced by each method in solving the 22 selected benchmark functions.

From Table 3, it is reported that the proposed CMPSOWV demonstrates the best search accuracy among all compared algorithms because it is able to obtain 18 best F_{mean} values out of the 22 selected benchmark functions. This is followed by the CVI-PSO, ETLBO, MABC, IPSO, ABC and CPSO-Shake that produce the best F_{mean} values in 12, 10, 8, 7, 7 and 6 tested functions, respectively. The proposed CMPSOWV is able to solve most tested functions and it is the only algorithm that can locate the global optima of G07, G14, G15, G18 and G19 functions. Although CMPSOWV is not able to reach the global optima of G05 and G21 functions, the F_{mean} values obtained for these two functions are the lowest among all compared methods. This indicates that the global best solutions obtained by CMPSOWV in these two functions are the closest to the theoretical values of global optima. All compared methods are able to find the global optima of G01, G08, G12 and G24 functions, suggesting that these four benchmarks functions are relatively easier to be solved as compared to the rest of tested problems.

4.5.2. Comparison of non-parametric statistical test results

Pairwise comparison between CMPSOWV and its peer algorithms are conducted using Wilcoxon signed rank test (García et al., 2008) and the results are summarized in Table 4 in terms of the R^+, R^-, *p* and *h* values. Significant improvements of CMPSOWV over the CPSO-Shake, IPSO, ETLBO, ABC and MABC are confirmed in Table 4 because their pairwise comparison results lead to *h*-values of "+". Meanwhile, no significant difference are observed between CMPSOWV and CVI-PSO in Table 4 as indicated by the *h*-value of "=".

Multiple comparisons analysis (Derrac et al., 2011; García et al., 2008) is also conducted to evaluate the performance of CMPSOWV further. The average rankings of all compared methods and their associated *p*-values derived from chi-square statistic are obtained from the Friedman test. In Table 5, all compared algorithms are ranked by Friedman test based on their search accuracy (i.e., F_{mean} values) as follows: CMPSOWV, ETLBO, CVI-PSO, MABC, IPSO, CPSO-Shake and ABC. The *p*-value of Friedman test is smaller than the threshold level of significance (i.e.,$\alpha = 0.05$), implying the existence of a significant global difference among all compared methods. Based on the Friedman test results, three post-hoc statistical analyses of Bonferroni-Dunn, Holm and Hochberg procedures are

Table 4
Wilcoxon signed rank test for the pairwise comparison between CMPSOWV with Six Peer algorithms.

CMPSOWV vs.	CPSO-Shake	IPSO	CVI-PSO	ELTBO	ABC	MABC
R^+	221.0	206.0	145.0	225.0	242.0	206.0
R^-	10.0	25.0	86.0	28.0	11.0	25.0
***p*-value**	2.29E-04	1.56E-03	2.97E-01	1.31E-03	1.66E-04	1.56E-03
***h*-value**	+	+	=	+	+	+

Table 5
Average ranking and associated *p*-values obtained through Friedman test.

Algorithm	CMPSOWV	CPSO-Shake	IPSO	CVI-PSO	ETLBO	ABC	MABC
Ranking	2.3180	4.8409	4.5227	3.7955	3.4091	4.9091	4.2045
Chi-Square Statistic	23.892857						
***p*-value**	**5.46E-04**						

Table 6
Adjusted *p*-Values Obtained for Bonferroni-Dunn, Holm and Hochberg Procedures.

CMPSOWV vs.	*z*	Unadjusted *p*	Bonferroni-Dunn *p*	Holm *p*	Hochberg *p*
ABC	3.98E+00	**7.00E-05**	**4.17E-04**	**4.17E-04**	**4.17E-04**
CPSO-Shake	3.87E+00	**1.07E-04**	**6.45E-04**	**5.37E-04**	**5.37E-04**
IPSO	3.38E+00	**7.13E-04**	**4.28E-03**	**2.85E-03**	**2.85E-03**
MABC	2.90E+00	**3.78E-03**	**2.27E-02**	**1.13E-02**	**1.13E-02**
CVI-PSO	2.27E+00	**2.33E-02**	1.40E-01	**4.67E-02**	**4.67E-02**
ETLBO	1.67E+00	9.40E-01	5.63E-01	9.40E-02	9.40E-02

conducted to detect the concrete differences with control algorithm of CMPSOWV (Derrac et al., 2011). Table 6 presents the *z*-values, unadjusted *p*-values and adjusted *p*-values (APVs) of these three post-hoc procedures. For$\alpha = 0.05$, all post-hoc procedures confirm the significant improvement of CMPSOWV over ABC, CPSO-Shake, IPSO and MABC because the APVs obtained are smaller than$\alpha = 0.05$. The significant performance differences between CMPSOWV and CVI-PSO are identified via the Holm and Hochberg procedures. While no significant performance difference is observed between CMPSOWV and ETLBO at $\alpha = 0.05$, both of the Holm and Hochberg procedures would confirm that CMPSOWV is significantly better than ETLBO if $\alpha = 0.10$.

4.6. Comparison between CMPSOWV and its variant

The impacts brought by the proposed modifications on CMPSOWV, i.e., the memory swarm evolution, multi-swarm technique performed on current swarm evolution and mutation scheme are investigated in this section. The performance of CMPSOWV is compared with a variant known as CPSOWV. The latter algorithm is not equipped with any of the aforementioned modifications and only Deb's rule is incorporated into the original PSOWV as the constraint handling technique to enable it solving COPs. Similar parameter settings are set for both CMPSOWV and CPSOWV to solve the 22 selected CEC 2006 benchmark problems.

Table 7 presents the F_{mean} and *SD* values obtained by both of CMPSOWV and CPSOWV in tackling the tested problems. The optimization performance of CMPSOWV completely dominates that of CPSOWV because the former algorithm is able to solve all benchmark problems with smaller or equal F_{mean} values. Without having the proposed modifications, CPSOWV can only locate the global optima of G08, G12 and G24 functions that are relatively easier to be solved. Evidently, the CPSOWV particles have higher tendency to be trapped into the local optima of most COPs as indicated by their inferior F_{mean} values. The comparison results in Table 7 prove that it is indeed crucial for the CMPSOWV to have multiple search operators with different exploration and exploitation strengths and diversity maintenance mechanisms in order to solve different COPs competitively. The pairwise comparison between CMPSOWV and CPSOWV are also conducted using the Wilcoxon signed rank test and the results are summarized in Table 8 in terms of R^+, R^-, *p* and *h* values. It is confirmed that the proposed CMPSOWV achieves significant performance improvement over CPSOWV because the *p*-value produced in the pairwise comparison is smaller than the predefined threshold of$\alpha = 0.05$.

4.7. Performance evaluation using CEC 2017 benchmark functions

In this section, the scalability of the proposed CMPSOWV in solving constrained optimization problems is further evaluated using another 28 benchmark functions introduced in CEC 2017 at the dimensional sizes of $D = 10$ and $D = 30$ (Wu, Mallipeddi & Suganthan, 2017). These CEC 2017 benchmark problems are more complicated than those of CEC 2006 because the problems developed in the former competition contain wider variety of constraints than the latter one. The detailed mathematical formulation of all CEC 2017 functions are presented in the technical report of Wu et al. (2017).

4.7.1. Algorithms comparisons and parameter settings

The proposed CMPSOWV is compared with another four constrained optimization algorithms, namely the constrained crow search algorithm (CCSA) (Askarzadeh, 2016), constrained grey wolf optimizer (CGWO) (Mirjalili, Mirjalili & Lewis, 2014), constrained simulated annealing (CSA) (Yang, 2014) and constrained water cycle algorithm (CWCA) (Eskandar, Sadollah, Bahreininejad & Hamdi, 2012). The parameter settings of all compared algorithms are adopted from the recommendation of their original literatures and summarized in Table 9. A method known as direct control of constraint was adopted by CCSA to abandon the infeasible solutions, while both of the CGWO and CSA employed penalty method to penalize any solutions that violate any constraints by assigning them with large objective function values. Similar with CMPSOWV, Deb's method was incorporated as the constraint handling technique of CWCA.

Table 7
Performance comparison between CMPSOWV and CPSOWV ($F_{mean} \pm SD$).

Function	G01	G02	G03	G04
CMPSOWV	**−15.0000**± 0.00E+00	**−0.803619**± 2.79E-07	**−1.000500**± 2.13E−11	**−30,665.5390**± 0.00E+00
CPSOWV	−9.1000± 4.52E+00	−0.778300± 2.04E−02	−0.050500± 1.15E−01	−30,521.0426± 2.30E+02
Function	G05	G06	G07	G08
CMPSOWV	**5126.4975**± 1.32E-01	**−6961.8140**± 0.00E+00	**24.3061**± 2.51E−11	**−0.095825**± 0.00E+00
CPSOWV	5383.5834± 2.95E+02	−5846.0271± 2.66E+03	26.2112± 1.38E+00	**−0.095825**± 2.85E−17
Function	G09	G10	G11	G12
CMPSOWV	**680.6300**± 2.76E−12	**7085.1718**± 7.13E+02	**0.74990**± 1.03E−12	**−1.0000**± 0.00E+00
CPSOWV	697.2289± 1.10E+01	20,932.5076± 3.80E+03	0.987460± 5.59E-02	**−1.0000**± 0.00E+00
Function	G13	G14	G15	G16
CMPSOWV	**0.434464**± 1.59E−01	**−47.76488**± 1.46E−07	**961.71502**± 4.63E−13	**−1.905155**± 3.48E−14
CPSOWV	1.042616± 3.48E−01	−40.98609± 2.56E+00	964.2845± 2.13E+00	−1.787484± 1.27E−01
Function	G17	G18	G19	G21
CMPSOWV	**8874.2046**± 3.64E−01	**−8.660254**± 3.64E−15	**32.655612**± 1.53E−05	**193.75354**± 2.52E-03
CPSOWV	9021.581± 1.24E+02	−0.800839± 1.02E−01	43.44555± 4.59E+00	203.89672± 1.32E+01
Function	G23	G24		
CMPSOWV	**−360.283699**± 1.43E+01	**−5.508013**± 0.00E+00		
CPSOWV	−325.96846± 5.66E+01	**−5.508013**± 1.28E−06		

Table 8
Wilcoxon Signed Rank Test for the Pairwise Comparison between CMPSOWV with CPSOWV.

Algorithm	R^+	R^-	p-value	h-value
CMPSOWV vs. CPSOWV	229.5	1.5	2.38E-06	+

For CEC 2017 benchmark functions, the users are free to set the population sizes of all compared algorithms while not exceeding the maximum fitness evaluation numbers (Wu et al., 2017). The population size of $N = 100$ is set for all compared algorithms to solve CEC 2017 benchmark functions at the dimension sizes of $D = 10$ and $D = 30$, except for CSA that employs the single solution for searching process, The tolerance of equality constraint is also set as $\delta = 0.0001$based on the recommendation of Liang et al. (2006). To ensure the fair performance comparisons, the maximum fitness evaluation numbers of all compared algorithms to solve each tested functions in 25 independent runs are set as Υ_{max}= 200,000 and Υ_{max}= 600,000 for the dimensional sizes of $D = 10$ and $D = 30$, respectively.

4.7.2. Comparison of f_{mean} *and* SD *results*

The F_{mean} and SD values obtained by all compared algorithms in solving all CEC 2017 benchmark functions at $D = 10$ and $D = 30$ are presented in Tables 10 and 11, respectively. From Table 10, it is observed that CMPSOWV is able to solve these 28 CEC 2017 benchmark functions with 20 best F_{mean} and three second best F_{mean} values for $D = 10$, implying the promising search accuracies of the proposed works in solving these challenging problems. Meanwhile, other peer algorithms of CSA, CGWO and CWCA are reported to be able to produce four, three and two best F_{mean} values in solving all CEC 2017 benchmarks at $D = 10$, suggesting the effectiveness of these methods in solving certain types of problems. Among all five compared algorithms, the search performances demonstrated by CCSA in solving the CEC 2017 benchmark functions are the most inferior because it fails to produce any best F_{mean} values.

When the dimensional size of all CEC 2017 benchmark problems are increased further to $D = 30$, some performance degradations are demonstrated by all compared algorithms in different extents as presented in Table 11. Despite of the increasing of problems' complexities, the proposed CMPSOWV still has the most competitive search accuracy for being able to produce 20 best F_{mean} and five second best F_{mean} values when it is used to tackle the 28 benchmark functions. For CSA, CGWO and CWCA, the similar search performances are observed because the number of F_{mean} values produced are three, five and one, respectively. Finally, it is also observed that the CCSA has the worst optimization performances in solving the CEC 2017 benchmark functions at $D = 30$.

4.7.3. Comparison of non-parametric statistical test results

Wilcoxon signed rank test is used for pairwise performance comparison between the proposed CMPSOWV and its competitors (García et al., 2008). The statistical analysis results in terms of the R^+, R^-, p and h values for the performance comparisons at $D = 10$ and $D = 30$ are summarized in the Tables 12 and 13, respectively. The significant performance improvements of CMPSOWV over the CCSA, CGWO, CSA and CWCA at $D = 10$ and $D = 30$ are verified by the Tables 12 and 13, respectively, because all pairwise comparison results produce the h-values of '+'.

The performance differences between the proposed CMPSOWV and its peer algorithms in solving the CEC 2017 benchmark functions are evaluated further using the multiple comparisons analysis (Derrac et al., 2011; García et al., 2008). The average rankings of all compared methods and their associated p-values derived from chi-square statistic in solving the benchmark problems at $D = 10$ and $D = 30$ are obtained using Friedman test and these results are summarized in Tables 14 and 15, respectively. For $D = 10$, all compared

Table 9
Parameter settings of all compared algorithms in solving CEC 2017 benchmark functions.

Algorithm	Parameter Settings
CMPSOWV	Population size $N = 100$, subpopulation size$N^k = 10$ with $k = 1,...,10$, acceleration coefficients $c_1 = c_2 = c_3 = 4.1/3$
CCSA	$N = 100$, awareness probability $AP = 0.1$, flight length $fl = 2$
CGWO	$N = 100$, parameter used to govern the fluctuation range$a$: $2.0 - 0.0$, effect of obstacles to approach prey in nature, $C \in [0, 2]$
CSA	Cooling factor for geometric cooling schedule$\alpha = 0.80$, initial temperature $T_0 = 1.0$, final temperature$T_f = 10^{-10}$, Boltzmann constant$k_B = 1.0$
CWCA	$N = 100$, number of rivers and sea$N_{sr} = 8$, evaporation condition constant$d_{max} = 10^{-5}$, parameter used to govern the stream's flow towards a specific river $C = 2.0$.

Table 10
Performance comparison between CMPSOWV with Four Peer algorithms in CEC 2017 Benchmark functions ($D = 10$).

Function	Criteria	CMPSOWV	CCSA	CGWO	CSA	CWCA
C01	F_{mean}	**1.77E-02**	1.67E+04	1.31E+01	2.63E+00	7.35E+00
	SD	1.60E-02	6.71E+03	2.83E+01	4.95E-01	9.30E+00
C02	F_{mean}	3.39E+00	1.45E+04	1.88E+01	**3.19E+00**	1.20E+01
	SD	8.22E-01	4.78E+03	4.08E+01	3.05E-01	1.43E+01
C03	F_{mean}	**1.20E-02**	1.71E+04	9.78E+01	3.18E+01	1.49E+04
	SD	1.41E-02	3.60E+03	9.21E+01	3.13E+01	2.24E+04
C04	F_{mean}	**1.37E+01**	2.39E+02	1.66E+01	8.03E+01	6.33E+01
	SD	1.06E-01	2.52E+01	8.32E+00	2.76E+01	1.92E+01
C05	F_{mean}	**4.70E-01**	3.82E+05	3.17E+00	1.16E+01	3.99E+00
	SD	3.02E-01	2.89E+05	5.50E-01	1.52E+00	4.30E-04
C06	F_{mean}	7.61E+01	6.69E+02	**1.75E+01**	1.70E+02	2.06E+03
	SD	2.50E+00	2.21E+02	1.25E+01	5.05E+01	6.31E+02
C07	F_{mean}	**−7.02E+01**	−3.51E+01	7.18E+01	−3.22E+01	5.50E+01
	SD	3.72E+01	8.59E+00	3.00E+02	2.60E+02	4.60E+01
C08	F_{mean}	**−1.21E-03**	3.41E+01	3.79E+00	6.63E+00	5.10E-03
	SD	2.28E-04	5.14E+00	5.19E+00	2.69E+00	8.41E-03
C09	F_{mean}	**−4.47E-03**	5.15E+00	2.80E+00	1.79E-02	2.40E+00
	SD	7.70E-04	1.77E+00	4.15E+00	7.72E-03	1.86E+00
C10	F_{mean}	**−5.09E-04**	8.03E+00	−4.68E-04	1.80E+01	3.77E-04
	SD	9.81E-06	1.85E+01	1.11E-04	8.57E+00	3.36E-04
C11	F_{mean}	**−3.37E+01**	1.77E+00	1.82E+01	1.66E+01	9.77E-01
	SD	9.63E+01	9.55E-01	4.84E+01	5.01E+01	2.11E+00
C12	F_{mean}	**4.44E+00**	8.39E+00	5.70E+00	7.26E+01	7.75E+00
	SD	5.41E-01	1.00E+00	1.89E+00	4.30E+00	7.41E+00
C13	F_{mean}	9.19E+00	3.18E+02	**3.64E+00**	6.39E+03	3.99E+00
	SD	1.43E-01	2.78E+02	3.16E+00	7.00E+03	1.36E-03
C14	F_{mean}	**1.13E+00**	2.10E+01	2.97E+00	1.01E+01	3.71E+00
	SD	1.34E-02	1.94E-01	4.59E-01	2.27E+00	2.68E-01
C15	F_{mean}	1.05E+01	3.51E+01	**−9.64E+00**	−7.18E+00	1.43E+01
	SD	4.76E+00	1.88E+01	1.27E-01	1.71E-01	4.10E+00
C16	F_{mean}	6.28E+01	3.04E+01	1.51E+01	**6.61E+00**	7.63E+01
	SD	9.29E+00	3.27E+00	3.30E+00	3.66E-01	1.09E+01
C17	F_{mean}	**9.85E-01**	4.89E+00	9.94E+00	1.17E+01	1.02E+00
	SD	3.42E-02	5.07E-01	8.38E-01	6.12E-01	2.60E-02
C18	F_{mean}	**1.62E+03**	1.36E+04	4.95E+03	7.09E+03	4.08E+03
	SD	2.623+03	4.58E+03	1.35E+03	8.64E+03	1.79E+03
C19	F_{mean}	**6.49E-01**	3.72E+01	1.33E+04	1.33E+04	1.97E+01
	SD	1.56E-01	3.01E+00	1.23E+01	2.39E+01	1.80E+01
C20	F_{mean}	**8.01E-01**	2.93E+00	1.31E+00	2.11E+00	2.01E+00
	SD	1.28E-01	5.60E-01	2.64E-01	1.00E-01	3.47E-01
C21	F_{mean}	**7.87E+00**	4.48E+02	1.25E+02	1.09E+02	1.04E+01
	SD	2.29E+00	1.08E+02	2.65E+02	2.35E+01	5.82E+00
C22	F_{mean}	**9.41E+01**	2.37E+03	2.30E+02	5.45E+03	3.19E+05
	SD	8.07E+01	1.30E+03	1.39E+02	2.59E+03	2.67E+05
C23	F_{mean}	**2.82E+00**	2.11E+01	3.35E+00	2.60E+01	3.63E+00
	SD	4.51E-01	7.34E-02	1.07E+00	1.10E+01	1.97E-01
C24	F_{mean}	5.68E+00	4.33E+01	1.23E+01	**−7.05E+00**	1.30E+01
	SD	4.66E-02	1.19E+01	8.60E+00	2.03E-02	3.58E+00
C25	F_{mean}	7.10E+01	5.30E+01	1.90E+01	**1.05E+01**	7.48E+01
	SD	8.98E+00	6.99E+00	1.15E+01	2.09E+00	7.08E+00
C26	F_{mean}	**1.01E+00**	1.18E+01	9.78E+00	1.25E+01	**1.01E+00**
	SD	5.99E-03	3.80E+00	8.96E-01	6.09E-01	9.16E-03
C27	F_{mean}	**4.18E+03**	3.89E+04	3.60E+04	2.27E+04	6.18E+03
	SD	4.20E+03	9.75E+03	2.11E+04	2.70E+04	4.50E+03
C28	F_{mean}	3.85E+01	4.33E+01	1.33E+04	1.33E+04	**3.46E+01**
	SD	1.03E+01	3.46E+00	1.78E+01	1.60E+01	9.86E+00
	w/t/l	–	26/0/2	23/0/5	23/0/5	25/1/2
	#BMF	20	0	3	4	2

algorithms are ranked by Friedman test based on their search accuracies as follows: CMPSOWV, CGWO, CWCA, CSA and CCSA. On the other hand, different rank values are obtained for all compared algorithms based on their respective F_{mean} values at $D = 30$, where: CMPSOWV, CWCA, CSA, CWGO and CCSA. The *p*-values of Friedman test reported in Tables 14 and 15 are smaller than the threshold level of significance (i.e.,$\alpha = 0.05$), suggesting the significant global difference among all compared methods at $D = 10$ and $D = 30$, respectively.

Referring to the Friedman test results, three post-hoc statistical analyses of Bonferroni-Dunn, Holm and Hochberg procedures are conducted to detect the concrete differences with control algorithm of CMPSOWV (Derrac et al., 2011). Tables 16 and 17 presents the *z*-values, unadjusted *p*-values and adjusted *p*-values (APVs) of these three post-hoc procedures at $D = 10$ and $D = 30$, respectively. The significant performance improvement of CMPSOWV over CCSA, CGWO, CSA and CWCA, in terms of the search accuracies in solving all CEC 2017 benchmark problems at $D = 10$ and $D = 30$, are confirmed by all post-hoc procedures for$\alpha = 0.05$.

4.8. Applications of CMPSOWV in engineering design problems

The proposed CMPSOWV is applied to solve four popular constrained engineering design problems known as welded beam design – case 1 (Ray & Liew, 2003), welded beam design – case 2

Table 11
Performance comparison between CMPSOWV with Four Peer algorithms in CEC 2017 Benchmark functions ($D = 30$).

Function	Criteria	CMPSOWV	CCSA	CGWO	CSA	CWCA
C01	F_{mean}	**2.48E-01**	1.27E+05	1.84E+03	3.24E+02	8.32E+03
	SD	3.56E-01	3.09E+04	4.23E+02	9.43E+01	4.65E+03
C02	F_{mean}	8.26E+02	1.39E+05	1.85E+03	**2.41E+02**	6.18E+03
	SD	3.53E+02	1.83E+04	4.84E+02	2.71E+01	3.15E+03
C03	F_{mean}	**8.45E+01**	2.25E+05	2.51E+03	3.77E+02	3.24E+05
	SD	1.18E+01	4.45E+04	1.94E+03	5.33E+01	9.13E+04
C04	F_{mean}	4.76E+02	9.34E+02	**1.02E+02**	5.19E+02	4.78E+02
	SD	6.82E+01	1.26E+02	2.93E+01	7.47E+01	3.94E+01
C05	F_{mean}	**3.01E-01**	4.40E+02	1.53E+03	1.60E+02	8.47E+01
	SD	3.58E-01	7.20E+01	7.24E+02	8.38E+01	5.39E+00
C06	F_{mean}	3.04E+02	3.21E+03	**1.48E+02**	6.75E+02	5.34E+03
	SD	5.38E+00	3.05E+02	6.40E+01	3.57E+02	6.71E+02
C07	F_{mean}	−3.84E+01	−2.44E+02	**−4.43E+02**	7.73E+01	−1.12E+01
	SD	9.04E+01	5.94E+01	1.34E+02	3.89E+02	2.52E+01
C08	F_{mean}	**−2.62E-04**	7.26E+01	1.34E+02	2.67E+02	2.92E+01
	SD	3.63E-05	1.78E+00	5.90E+01	1.51E+02	1.04E+01
C09	F_{mean}	**2.45E-03**	9.76E+00	4.17E+01	1.56E+01	1.90E+01
	SD	3.42E-04	8.38E-01	1.80E+01	3.17E+00	4.27E-01
C10	F_{mean}	**−1.29E-04**	6.17E+01	8.11E+03	4.16E+02	3.34E+00
	SD	6.65E-05	6.57E+00	4.82E+03	3.74E+01	2.01E+00
C11	F_{mean}	**−3.18E+02**	−8.28E+01	6.85E+01	2.93E+02	−3.05E+00
	SD	1.05E+03	1.63E+01	8.03E+01	4.48E+02	1.78E+01
C12	F_{mean}	**6.18E+00**	7.61E+04	1.52E+03	4.47E+02	9.78E+00
	SD	2.57E+00	4.01E+03	9.38E+02	2.49E+01	8.91E-04
C13	F_{mean}	**1.47E+02**	3.61E+06	3.01E+07	7.54E+04	8.73E+04
	SD	5.19E+01	7.48E+05	5.98E+07	1.50E+04	1.95E+05
C14	F_{mean}	**2.50E+00**	2.14E+01	7.99E+02	1.43E+02	7.95E+00
	SD	2.88E-01	8.95E-02	5.91E+02	3.40E+01	5.88E-01
C15	F_{mean}	2.31E+01	8.61E+01	8.00E+00	**−1.41E+00**	2.70E+01
	SD	8.40E+00	3.58E+00	3.46E+00	2.63E+00	1.66E+00
C16	F_{mean}	1.51E+02	1.15E+03	6.02E+01	**3.94E+01**	2.25E+02
	SD	1.99E+01	6.85E+01	1.86E+01	4.95E+00	2.86E+01
C17	F_{mean}	**1.07E+00**	1.95E+01	1.07E+03	3.24E+01	2.02E+00
	SD	1.63E-01	1.93E+00	8.45E+02	4.49E-01	6.51E-01
C18	F_{mean}	**5.67E+03**	6.93E+04	2.88E+06	1.07E+05	7.30E+03
	SD	7.08E+03	9.07E+03	5.52E+06	5.87E+04	3.22E+03
C19	F_{mean}	**1.94E+00**	1.25E+02	4.29E+04	4.31E+04	6.53E+01
	SD	2.23E-01	5.42E+00	1.85E+01	1.10E+01	9.59E+00
C20	F_{mean}	**2.98E+00**	1.13E+01	4.89E+00	8.99E+00	3.78E+00
	SD	1.94E-01	1.65E-01	2.19E+00	3.34E-01	6.36E-01
C21	F_{mean}	**4.21E+01**	2.46E+05	6.10E+03	736E+02	4.40E+01
	SD	1.93E+01	3.30E+04	4.09E+03	4.45E+01	3.44E+01
C22	F_{mean}	5.13E+06	6.02E+07	2.48E+07	**2.05E+06**	1.49E+07
	SD	2.24E+06	2.33E+07	3.40E+07	1.93E+06	1.10E+07
C23	F_{mean}	**2.49E+00**	2.14E+01	5.95E+03	5.33E+02	3.49E+00
	SD	1.30E-01	3.39E-02	5.73E+03	1.33E+02	4.41E-01
C24	F_{mean}	1.93E+01	1.61E+02	4.41E+02	**2.68E+00**	2.25E+01
	SD	1.62E+00	1.90E+01	3.10E+02	7.00E-01	4.76E+00
C25	F_{mean}	**1.73E+00**	2.20E+03	9.33E+02	7.94E+01	3.74E+02
	SD	1.05E-01	1.96E+02	1.14E+03	7.26E+00	8.61E+01
C26	F_{mean}	**1.03E+00**	6.25E+01	3.50E+03	2.01E+02	**1.03E+00**
	SD	1.81E-03	1.15E+01	3.28E+03	7.75E+01	6.46E-04
C27	F_{mean}	**5.26E+03**	2.22E+05	9.84E+06	2.15E+06	5.45E+03
	SD	8.85E+01	4.56E+04	1.08E+07	2.07E+06	2.82E+03
C28	F_{mean}	**1.39E+02**	1.78E+02	4.30E+04	4.31E+04	1.82E+02
	SD	9.58E+00	6.80E+00	2.90E+01	1.22E+01	2.08E+01
	w/t/l	–	27/0/1	23/0/5	23/0/5	27/1/0
	#BMF	20	0	3	5	1

Table 12
Wilcoxon signed rank test for the pairwise comparison between CMPSOWV with Four Peer algorithms at $D = 10$.

CMPSOWV vs.	CCSA	CGWO	CSA	CWCA
R^+	386.0	324.0	348.0	351.0
R^-	20.0	82.0	58.0	27.0
***p*-value**	2.90E-05	5.66E-03	9.22E-04	9.50E-05
***h*-value**	+	+	+	+

Table 13
Wilcoxon signed rank test for the pairwise comparison between CMPSOWV with Four Peer algorithms at $D = 30$.

CMPSOWV vs.	CCSA	CGWO	CSA	CWCA
R^+	393.0	378.0	343.0	378.0
R^-	13.0	28.0	63.0	0.0
***p*-value**	1.40E-05	6.40E-05	1.38E-03	5.00E-06
***h*-value**	+	+	+	+

Table 14
Average ranking and associated *p*-values obtained through Friedman test ($D = 10$).

Algorithm	CMPSOWV	CCSA	CGWO	CSA	CWCA
Ranking	1.5179	4.1786	2.8929	3.3929	3.0179
Chi-Square Statistic	42.021429				
***p*-value**	**0.00E+00**				

Table 15
Average ranking and associated *p*-values obtained through Friedman test ($D = 30$).

Algorithm	CMPSOWV	CCSA	CGWO	CSA	CWCA
Ranking	1.4107	3.7500	3.7500	3.2143	2.8750
Chi-Square Statistic	41.578571				
***p*-value**	**0.00E+00**				

(Huang, Wang & He, 2007), speed reducer design (Ray & Liew, 2003) and pressure vessel design (Huang et al., 2007) in order to further evaluate its performance in addressing real-world applications. The problem formulation of each engineering design problem are presented in Appendix A.

4.8.1. Welded beam design – case 1

The welded beam design problem – case 1 (Ray & Liew, 2003) is a single objective constrained optimization problem that aims to minimize the total cost of beam while satisfying the technical constraints such as bending stress, buckling load, shear stress and end deflection. Four continuous design variables known as the thickness of beam, width of beam, length of the weld and thickness of weld needs to be optimized in this constrained engineering problems.

The optimization performance of CMPSOWV in solving the welded beam design – case 1 is compared with, society and civilization algorithm (SCM) (Ray & Liew, 2003), dynamic stochastic selection modified differential evolution (DSS-MDE) (Zhang, Luo & Wang, 2008), accelerating adaptive trade-off model (AATM) (Wang, Cai & Zhou, 2009), differential evolution with level comparison (DELC) (L. Wang & Li, 2010), and improved differential evolution with ranking-based mutation and multiple trial vector generation (Rank-iMDDE) (Gong, Cai & Liang, 2014). The F_{mean} and *SD* values achieved by the selected peer algorithms in solving this real-world application are extracted from their corresponding literatures. From Table 18, it is observed that proposed CMPSOWV is able to solve the welded beam design – case 1 with the lowest F_{mean} value together with other peer algorithms such as DSS-MDE, DELC and Rank-iMDDE, implying the promising search accuracy of these compared methods.

4.8.2. Welded beam design – case 2

The welded beam design problem – case 2 (Huang et al., 2007) also aims to minimize the total cost of a welded beam subjected to the constraints of bending stress, buckling load, shear stress, end deflection and side constraints. Compared to case 1, two additional constraints are incorporated into the welded beam design problem – case 2 to increase the problem complexity. Four continuous design variables known as thickness of beam, width of beam, length of the weld and thickness of weld are to be optimized in this constrained engineering problems.

The optimization performance of CMPSOWV in solving the welded beam design – case 2 is compared with the co-evolutionary differential evolution (CO-DE) (Huang et al., 2007), constraint violation with interval arithmetic PSO (CVI-PSO) (Mazhoud et al., 2013), multi-population genetic algorithm with automatic dynamic penalization (BIANCA) (Montemurro, Vincenti & Vannucci, 2013), multi-view differential evolution (MVDE) (de Melo & Carosio, 2013) and improved differential evolution with ranking-based mutation and multiple trial vector generation (Rank-iMDDE) (Gong et al., 2014). Table 19 shows the comparison results of F_{mean} and *SD* for the welded beam design – case 2. It is reported that only two compared algorithms, i.e., CMPSOWV and Rank-iMDDE are able to locate the global optimum of this challenging engineering design problem. Furthermore, the proposed CMPSOWV is able to achieve the smallest *SD* values, implying its promising consistency to acquire the optimal solutions of this problem in all simulation runs.

4.8.3. Speed reducer design

The objective function of speed reducer design problem (Ray & Liew, 2003) is the weight minimization subjected to the technical constraints such as surface stress, stresses in the shaft, bending stress of the gear teeth and transverse deflections of the shafts. Seven continuous design parameters known as the face width, module of teeth, number of teeth in the pinion, length of the first shaft between bearings, length of the second shaft between bearings and the diameter of first and second shafts need to be optimized.

The F_{mean} and *SD* values produced by CMPSOWV in solving the speed reducer design problem are compared with those of the society and civilization algorithm (SCM) (Ray & Liew, 2003), multiple trial vectors in differential evolution (MDDE) (Mezura-Montes, Coello, Velázquez-Reyes & Muñoz-Dávila, 2007), dynamic stochastic selection modified differential evolution (DSS-MDE) (Zhang et al., 2008), accelerating adaptive trade-off model (AATM) (Wang et al., 2009), differential evolution with level comparison (DELC) (Wang & Li, 2010), multi-view differential evolution (MVDE) (de Melo & Carosio, 2013) and improved differential evolution with ranking-based mutation and multiple trial vector generation (Rank-iMDDE) (Gong et al., 2014). The comparison results for speed reducer design problems are summarized in Table 20. Accordingly, CMPSOWV is one of the five compared methods that can locate the global optimum successfully. The *SD* value of CMPSOWV is also the most competitive, suggesting its consistency to solve this problem with promising search accuracy.

4.8.4. Pressure vessel design

The objective of solving pressure vessel design problem (Huang et al., 2007) is to minimize the total cost that consists of the material, forming and welding costs. There are four design variables need to be optimized in this problem, namely the thickness of shell, thickness of head, inner radius and length of cylindrical section of the vessel, respectively. Notably, the first two design variables are the integer multiples of 0.0625 inches (i.e., the available thicknesses of rolled steel plates), while last two design variables have the continuous values.

Table 16
Adjusted *p*-values obtained for Bonferroni-Dunn, Holm and Hochberg procedures ($D = 10$).

CMPSOWV vs.	*z*	Unadjusted *p*	Bonferroni-Dunn *p*	Holm *p*	Hochberg *p*
CCSA	6.296399	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
CSA	4.437060	**9.00E-06**	**3.60E-05**	**2.70E-05**	**2.70E-05**
CWCA	3.549648	**3.86E-04**	**1.54E-03**	**7.71E-04**	**7.71E-04**
CGWO	3.253844	**1.14E-03**	**4.55E-03**	**1.14E-03**	**1.14E-03**

Table 17
Adjusted p-values obtained for Bonferroni-Dunn, Holm and Hochberg procedures ($D = 30$).

CMPSOWV vs.	z	Unadjusted p	Bonferroni-Dunn p	Holm p	Hochberg p
CCSA	5.535760	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
CSA	5.535760	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
CWCA	4.268029	**2.00E-05**	**7.90E-05**	**3.90E-05**	**3.90E-05**
CGWO	3.465132	**5.30E-04**	**2.12E-03**	**5.30E-04**	**5.30E-04**

Table 18
Performance comparison of Welded beam design – Case 1.

	CMPSOWV	SCM	DSS-MDE	AATM	DELC	Rank-iMDDE
F_{mean}	**2.3809565**	3.0025883	**2.3809565**	2.3869762	**2.3809565**	**2.3809565**
SD	3.19E−09	9.60E−01	3.19E−10	2.2E −03	2.60E −12	**7.18E −14**

Table 19
Performance comparison of Welded beam design – Case 2.

	CMPSOWV	CO-DE	CVI-PSO	BIANCA	MVDE	Rank-iMDDE
F_{mean}	**1.724852309**	1.768185	1.725124	1.752201	1.7248621	**1.724852309**
SD	**2.35E−13**	2.22E −02	6.12E −04	2.30E-02	7.88E −06	7.71E −11

Table 20
Performance comparison of speed reducer design.

	CMPSOWV	SCM	MDDE	DSS-MDE	AATM	DELC	MVDE	ICDE
F_{mean}	**2994.471066**	3001.758264	2996.367	**2994.471066**	2994.585417	**2994.471066**	**2994.471066**	**2994.471066**
SD	**1.14E−13**	4.00E+00	8.20E −03	3.58E −12	3.30E −02	1.90E−12	2.82E −07	7.93E −13

Table 21
Performance comparison of pressure vessel design.

	CMPSOWV	CO-DE	DELC	COMDE	CVI-PSO	MVDE	BIANCA	Rank-iMDDE
F_{mean}	**6059.714335**	6085.2303	**6059.714335**	**6059.714335**	6292.1231	6059.997236	6182.0022	**6059.714335**
SD	**2.14E-13**	4.30E + 01	2.10E −11	3.62E −10	2.88E + 02	2.91E + 00	1.22E+02	7.57E-07

The optimization results of pressure vessel design produced by CMPSOWV are compared with those of the co-evolutionary differential evolution (CO-DE) (Huang et al., 2007), differential evolution with level comparison (DELC) (Wang & Li, 2010), constrained optimization based on modified differential evolution algorithm (COMDE) (Mohamed & Sabry, 2012), constraint violation with interval arithmetic PSO (CVI-PSO) (Mazhoud et al., 2013), multi-view differential evolution (MVDE) (de Melo & Carosio, 2013), multi-population genetic algorithm with automatic dynamic penalization (BIANCA) (Montemurro et al., 2013), and improved differential evolution with ranking-based mutation and multiple trial vector generation (Rank-iMDDE) (Gong et al., 2014). Table 21 presents the F_{mean} and SD values produced by all compared algorithms. Three out of six compared algorithms, including CMPSOWV, are able to find the global optimum of pressure vessel design problem. The competitive consistency of CMPSOWV in solving this problem can also be observed from its lowest SD value.

5. Conclusion

In this paper, a constrained multi-swarm particle swarm optimization without velocity (CMPSOWV) is proposed to solve constrained optimization problems (COPs) more effectively. A constraint handling method of Deb's rule is incorporated to guide the CMPSOWV population searching towards the feasible regions of search space before optimizing the objective functions within feasible regions. The multiple search operator concept is also introduced via the current swarm evolution and memory swarm evolution of CMPSOWV in order to improve its robustness to tackle different types of COPs. Two diversity maintenance schemes known as the multi-swarm technique and mutation scheme are applied to the current swarm evolution and the global best particle, respectively, to prevent the premature convergence of algorithm. Extensive simulation studies were conducted to evaluate the performance of CMPSOWV and it is proven that the proposed algorithm has demonstrated promising search accuracy to solve the selected benchmark functions and constrained engineering design problems.

In the future studies, the concept of multiple constraint handling techniques can be incorporated into the CMPSOWV to improve its robustness in solving different types of COPs more effectively. An adaptive selection scheme can be developed to enable each CMPSOWV particle to select the most suitable constraint handling technique to solve a given COP based on information such as current fitness and constraint violation of particle. It is also worth to investigate the modifications required to be introduced into CMPSOWV so that the proposed algorithm can be used to solve other types of problems such as multi-objective optimization problems, multi-modal optimization problem, dynamic optimization problems and etc.

Declaration of competing interest

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

Credit authorship contribution statement

Koon Meng Ang: Software, Formal analysis, Writing - original draft, Visualization. **Wei Hong Lim:** Conceptualization,

Methodology, Validation, Writing - original draft, Supervision. **Nor Ashidi Mat Isa:** Conceptualization, Methodology, Validation, Writing - review & editing, Supervision. **Sew Sun Tiang:** Software, Formal analysis, Writing - review & editing, Visualization. **Chin Hong Wong:** Software, Formal analysis, Writing - review & editing, Visualization.

Acknowledgment

This work is partially supported by the Fundamental Research Grant Scheme(FRGS) with project code of FRGS/1/2018/TK04/USM/01/2 and UCSI University Pioneer Scientist Incentive Fund (PSIF) with project code of Proj-In-FETBE-50.

Appendix A: Constrained Engineering Design Problems

A.1. Welded beam design – Case 1

Minimize :

$$f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

Subject to:

$$g_1(\vec{x}) = \tau(\vec{x}) - \tau_{\max} \le 0$$
$$g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{\max} \le 0$$
$$g_3(\vec{x}) = x_1 - x_4 \le 0$$
$$g_4(\vec{x}) = \delta(\vec{x}) - \delta_{\max} \le 0$$
$$g_5(\vec{x}) = P - P_c(\vec{x}) \le 0$$
$$0.125 \le x_1 \le 10.0;\ 0.1 \le x_2 \le 10.0;\ 0.1 \le x_3 \le 10;\ 0.1 \le x_4 \le 10.0$$

Where:

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + \frac{2\tau'\tau''x^2}{2R} + (\tau'')^2};$$
$$\tau' = \frac{P}{\sqrt{2}x_1x_2};$$
$$\tau'' = \frac{MR}{J}$$
$$M = P\left(L + \frac{x_2}{2}\right);$$
$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2};$$
$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\};$$
$$\sigma(\vec{x}) = \frac{6PL}{x_4x_3^2};$$
$$\delta(\vec{x}) = \frac{4PL^3}{Ex_3^3x_4};$$
$$P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right);$$
$$P = 6000\ lb,\ L = 14\ in,\ \delta_{\max} = 0.25\ in,\ E = 30 \times 10^6\ psi,$$
$$G = 12 \times 10^6\ psi,\ \tau_{\max} = 13600\ psi,\ \sigma_{\max} = 30000\ psi.$$

A.2. Welded beam design – Case 2

Minimize :

$$f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$$

Subject to:

$$g_1(\vec{x}) = \tau(\vec{x}) - \tau_{\max} \le 0;$$
$$g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{\max} \le 0;$$
$$g_3(\vec{x}) = x_1 - x_4 \le 0;$$
$$g_4(\vec{x}) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0;$$
$$g_5(\vec{x}) = 0.125 - x_1 \le 0;$$
$$g_6(\vec{x}) = \delta(\vec{x}) - \delta_{\max} \le 0;$$
$$g_7(\vec{x}) = P - P_c(\vec{x}) \le 0;$$
$$0.1 \le x_1 \le 2,\ 0.1 \le x_2 \le 10,\ 0.1 \le x_3 \le 10,\ 0.1 \le x_4 \le 2;$$

Where:

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2};$$
$$\tau' = \frac{P}{\sqrt{2}x_1x_2};$$
$$\tau'' = \frac{MR}{J};$$
$$M = P\left(L + \frac{x_2}{2}\right);$$
$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2};$$
$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\};$$
$$\sigma(\vec{x}) = \frac{6PL}{x_4x_3^2};$$
$$\delta(\vec{x}) = \frac{4PL^3}{Ex_3^3x_4};$$
$$P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right);$$
$$P = 6000\ lb,\ L = 14\ in,\ \delta_{\max} = 0.25\ in,\ E = 30 \times 10^6\ psi,$$
$$G = 12 \times 10^6\ psi,\ \tau_{\max} = 13600\ psi,\ \sigma_{\max} = 30000\ psi.$$

A.3. Speed reducer design

Minimize :

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

Subjectto :

$$g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \le 0;$$
$$g_2(\vec{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0;$$
$$g_3(\vec{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0;$$
$$g_4(\vec{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0;$$
$$g_5(\vec{x}) = \frac{\left[\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6\right]^{1/2}}{110.0x_6^3} - 1 \le 0;$$
$$2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, 7.3 \le x_4 \le 8.3;$$
$$7.3 \le x_5 \le 8.3, 2.9 \le x_6 \le 3.9, 5.0 \le x_7 \le 5.5;$$

A.4. Pressure vessel design

Minimize :

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

Subject to:

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

References

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

Banks, A., Vincent, J., & Anyakoha, C. (2007). A review of particle swarm optimization. part I: Background and development. *Natural Computing, 6*, 467–484.

Banks, A., Vincent, J., & Anyakoha, C. (2008). A review of particle swarm optimization. part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing, 7*, 109–124.

Ben Guedria, N. (2016). Improved accelerated pso algorithm for mechanical engineering optimization problems. *Applied Soft Computing, 40*, 455–467.

Benhamou, F., & Older, W. J. (1997). Applying interval arithmetic to real, integer, and boolean constraints. *The Journal of Logic Programming, 32*, 1–24.

Cagnina, L. C., Esquivel, S. C., & Coello, C. A. C. (2011). Solving constrained optimization problems with a hybrid particle swarm optimization algorithm. *Engineering Optimization, 43*, 843–866.

Clerc, M., & Kennedy, J. (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation, 6*, 58–73.

Coelho, L. d. S. (2010). Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Systems with Applications, 37*, 1676–1683.

Daneshyari, M., & Yen, G. G. (2012). Constrained multiple-swarm particle swarm optimization within a cultural framework. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 42*, 475–490.

Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution – An updated survey. *Swarm and Evolutionary Computation, 27*, 1–30.

Das, S., & Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation, 15*, 4–31.

Davoodi, E., Hagh, M. T., & Zadeh, S. G. (2014). A hybrid improved quantum-behaved particle swarm optimization–simplex method (IQPSOS) to solve power system load flow problems. *Applied Soft Computing, 21*, 171–179.

de Melo, V. V., & Carosio, G. L. C. (2013). Investigating multi-view differential evolution for solving constrained engineering design problems. *Expert Systems with Applications, 40*, 3370–3377.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering, 186*, 311–338.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation, 1*, 3–18.

Elsayed, S. M., Sarker, R. A., & Mezura-Montes, E. (2014). Self-adaptive mix of particle swarm methodologies for constrained optimization. *Information Sciences, 277*, 216–233.

El-Sherbiny, M. M. (2011). Particle swarm inspired optimization algorithm without velocity equation. *Egyptian Informatics Journal, 12*, 1–8.

Eskandar, H., Sadollah, A., Bahreininejad, A., & Hamdi, M. (2012). Water cycle algorithm – A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers & Structures, 110-111*, 151–166.

García, S., Molina, D., Lozano, M., & Herrera, F. (2008). A study on the use of non–parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics, 15*, 617.

Gong, W., Cai, Z., & Liang, D. (2014). Engineering optimization by means of an improved constrained differential evolution. *Computer Methods in Applied Mechanics and Engineering, 268*, 884–904.

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

Hernandez Aguirre, A., Muñoz Zavala, A., & Villa Diharce, E. B. R. S. (2007). COPSO: Constrained optimization via PSO algorithm. *Center for Research in Mathematics (CIMAT)*.

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

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. *Journal of Global Optimization, 39*, 459–471.

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

Krohling, R. A., & Coelho, L. d. S. (2006). Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 36*, 1407–1416.

Lan, R., Zhang, L., Tang, Z., Liu, Z., & Luo, X. (2019). A hierarchical sorting swarm optimizer for large-scale optimization. *IEEE Access, 7*, 40625–40635.

Lawler, E. L., & Wood, D. E. (1966). Branch-and-bound methods: A survey. *Operations Research, 14*, 699–719.

Li, C., Yang, S., & Nguyen, T. T. (2012). A self-learning particle swarm optimizer for global optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 42*, 627–646.

Liang, J. J., Runarsson, T. P., Mezura-Montes, E., Clerc, M., Suganthan, P. N., Coello, C. A. C., et al. (2006). Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. *Nanyang Technological University*.

Lim, W. H., & Isa, N. A. M. (2015). Particle swarm optimization with dual-level task allocation. *Engineering Applications of Artificial Intelligence, 38*, 88–110.

Lim, W. H., Isa, N. A. M., Tiang, S. S., Tan, T. H., Natarajan, E., & Wong, C. H. (2018). A self-adaptive topologically connected-based particle swarm optimization. *IEEE Access, 6*, 65347–65366.

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

Liu, H.-. R., Cui, J.-. C., Lu, Z.-. D., Liu, D.-. Y., & Deng, Y.-. J. (2019). A hierarchical simple particle swarm optimization with mean dimensional information. *Applied Soft Computing, 76*, 712–725.

Liu, Z., Li, Z., Zhu, P., & Chen, W. (2018). A parallel boundary search particle swarm optimization algorithm for constrained optimization problems. *Structural and Multidisciplinary Optimization, 58*, 1505–1522.

Machado-Coelho, T. M., Machado, A. M. C., Jaulin, L., Ekel, P., Pedrycz, W., & Soares, G. L. (2017). An interval space reducing method for constrained problems with particle swarm optimization. *Applied Soft Computing, 59*, 405–417.

Mazhoud, I., Hadj-Hamou, K., Bigeon, J., & Joyeux, P. (2013). Particle swarm optimization for solving engineering problems: A new constraint-handling mechanism. *Engineering Applications of Artificial Intelligence, 26*, 1263–1273.

Mezura-Montes, E., & Cetina-Domínguez, O. (2012). Empirical analysis of a modified artificial Bee colony for constrained numerical optimization. *Applied Mathematics and Computation, 218*, 10943–10973.

Mezura-Montes, E., & Coello Coello, C. A. (2011). Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation, 1*, 173–194.

Mezura-Montes, E., Coello Coello, C. A., Velázquez-Reyes, J., & Muñoz-Dávila, L. (2007). Multiple trial vectors in differential evolution for engineering design. *Engineering Optimization, 39*, 567–589.

Mezura-Montes, E., & Flores-Mendoza, J. I. (2009). Improved particle swarm optimization in constrained numerical search spaces. In R. Chiong (Ed.), *Nature-Inspired algorithms for optimisation* (pp. 299–332). Berlin Heidelberg: Springer.

Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary algorithms for constrained parameter optimization problems. *Evol. Comput., 4*, 1–32.

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

Mistry, K., Zhang, L., Neoh, S. C., Lim, C. P., & Fielding, B. (2017). A micro-ga embedded PSO feature selection approach to intelligent facial emotion recognition. *IEEE Transactions on Cybernetics, 47*, 1496–1509.

Mohamed, A. W., & Sabry, H. Z. (2012). Constrained optimization based on modified differential evolution algorithm. *Information Sciences, 194*, 171–208.

Mohanty, C. P., Mahapatra, S. S., & Singh, M. R. (2016). A particle swarm approach for multi-objective optimization of electrical discharge machining process. *Journal of Intelligent Manufacturing, 27*, 1171–1190.

Montemurro, M., Vincenti, A., & Vannucci, P. (2013). The Automatic Dynamic Penalisation method (ADP) for handling constraints with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering, 256*, 70–87.

Quaranta, G., Marano, G. C., Greco, R., & Monti, G. (2014). Parametric identification of seismic isolators using differential evolution and particle swarm optimization. *Applied Soft Computing, 22*, 458–464.

Rao, R., & Patel, V. (2012). An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems. *International Journal of Industrial Engineering Computations, 3*, 535–560.

Ray, T., & Liew, K. M. (2003). Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation, 7*, 386–396.

Runarsson, T. P., & Xin, Y. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation, 4*, 284–294.

Solihin, M. I., Wahyudi, & Akmeliawati, R. (2009). PSO-based optimization of state feedback tracking controller for a flexible link manipulator. In *2009 International Conference of Soft Computing and Pattern Recognition* (pp. 72–76).

Sun, C.-l., Zeng, J.-c., & Pan, J.-s. (2011). An improved vector particle swarm optimization for constrained optimization problems. *Information Sciences, 181*, 1153–1163.

Tam, J. H., Ong, Z. C., Ismail, Z., Ang, B. C., & Khoo, S. Y. (2019). A new hybrid GA–ACO–PSO algorithm for solving various engineering design problems. *International Journal of Computer Mathematics, 96*, 883–919.

Tang, S. H., Ang, C. K., Ariffin, M. K. A. B. M., & Mashohor, S. B. (2014). Predicting the motion of a robot manipulator with unknown trajectories based on an artificial neural network. *International Journal of Advanced Robotic Systems, 11*, 176.

Tang, S. H., Ang, C. K., Nia, D. N., Ariffin, M. K. A. M., & Khaksar, W. (2013). Planning for redundant manipulator based on back-propagation neural network. *Advanced Science Letters, 19*, 3307–3310.

Valle, Y. d., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J., & Harley, R. G. (2008). Particle swarm optimization: Basic concepts, variants and applications in power systems. *IEEE Transactions on Evolutionary Computation, 12*, 171–195.

van den Bergh, F., & Engelbrecht, A. P. (2006). A study of particle swarm optimization particle trajectories. *Information Sciences, 176*, 937–971.

Van, M., & Kang, H. (2016). Bearing defect classification based on individual wavelet local fisher discriminant analysis with particle swarm optimization. *IEEE Transactions on Industrial Informatics, 12*, 124–135.

Vrugt, J. A., Robinson, B. A., & Hyman, J. M. (2009). Self-adaptive multimethod search for global optimization in real-parameter spaces. *IEEE Transactions on Evolutionary Computation, 13*, 243–259.

Wang, H. (2012). Opposition-based barebones particle swarm for constrained nonlinear optimization problems. *Mathematical Problems in Engineering, 2012*, 12.

Wang, J., & Yin, Z. (2008). A ranking selection-based particle swarm optimizer for engineering design optimization problems. *Structural and Multidisciplinary Optimization, 37*, 131–147.

Wang, L., & Li, L.-p (2010). An effective differential evolution with level comparison for constrained engineering design. *Structural and Multidisciplinary Optimization, 41*, 947–963.

Wang, Y., Cai, Z., & Zhou, Y. (2009). Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization. *International Journal for Numerical Methods in Engineering, 77*, 1501–1534.

Wang, Y., Li, B., Weise, T., Wang, J., Yuan, B., & Tian, Q. (2011). Self-adaptive learning based particle swarm optimization. *Information Sciences, 181*, 4515–4538.

Wood, G. R. (1991). Multidimensional bisection applied to global optimisation. *Computers & Mathematics with Applications, 21*, 161–172.

Wu, G., Mallipeddi, R., & Suganthan, P.N. (.2017). Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization. In *Technical Report*. Changsha, Hunan, P.R. China: National University of Defense Technology.

Xu, G., Cui, Q., Shi, X., Ge, H., Zhan, Z.-. H., & Lee, H. P. (2019). Particle swarm optimization based on dimensional learning strategy. *Swarm and Evolutionary Computation, 45*, 33–51.

Xu, J. J., & Xin, Z. H. (2005). An extended particle swarm optimizer. In *19th IEEE International Parallel and Distributed Processing Symposium* (pp. 1–5).

Yang, X.-. S. (2014). Chapter 4- Simulated Annealing. In X.-S. Yang (Ed.), *Nature-Inspired optimization algorithms* (pp. 67–75). Oxford: Elsevier.

Yao, L., Damiran, Z., & Lim, W. H. (2017). Energy management optimization scheme for smart home considering different types of appliances. In *2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)* (pp. 1–6).

Yao, L., Lai, C., & Lim, W. H. (2015). Home energy management system based on photovoltaic system. In *2015 IEEE International Conference on Data Science and Data Intensive Systems* (pp. 644–650).

Zahara, E., & Kao, Y.-. T. (2009). Hybrid nelder–mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications, 36*, 3880–3886.

Zavala, A. E. M., Aguirre, A. H., & Diharce, E. R. V. (2005). Particle evolutionary swarm optimization algorithm (PESO). In *Proceedings of the Sixth Mexican International Conference on Computer Science* (pp. 282–289). IEEE Computer Society.

Zhang, M., Luo, W., & Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences, 178*, 3043–3074.

Zhao, J., Wen, F., Dong, Z. Y., Xue, Y., & Wong, K. P. (2012). Optimal dispatch of electric vehicles and wind power using enhanced particle swarm optimization. *IEEE Transactions on Industrial Informatics, 8*, 889–899.

Zou, F., Chen, D., & Xu, Q. (2019). A survey of teaching–learning-based optimization. *Neurocomputing, 335*, 366–383.