Journal of Computational Science 41 (2020) 101086

Contents lists available at ScienceDirect

Journal of Computational Science

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

Sensitivity analysis of control parameters in particle swarm optimization

Mewael Isiet [a,*], Mohamed Gadala [b,c]

[a] Norman B. Keevil Institute of Mining Engineering, 511-6350 Stores Road, Vancouver, BC, V6T 1Z4, Canada
[b] Mechanical Engineering Department, University of British Columbia, 2054-6250 Applied Science Lane, Vancouver, BC, V6T 1Z4, Canada
[c] Mechanical Engineering Department, Abu Dhabi University, Abu Dhabi, United Arab Emirates

ARTICLE INFO

Article history:
Received 2 January 2019
Received in revised form 13 August 2019
Accepted 28 January 2020
Available online 3 February 2020

Keywords:
Particle swarm optimization
Sensitivity analysis
Control parameters
Constrained optimization
Optimal parameter set
Metaheuristics

ABSTRACT

Particle Swarm Optimization (PSO) is a powerful nature-inspired metaheuristic optimization method that may determine optimal solutions of engineering problems in fewer evaluations compared to other optimization methods. However, the literature shows that PSO may suffer from converging prematurely to a local solution, and this occurs due to poor tuning of the control parameters in PSO. In this paper, an extensive parametric sensitivity analysis was conducted to understand the impact of the individual control parameters and their respective influence on the performance of PSO. A benchmark constrained optimization problem was considered for studying PSO by modifying each parameter one-at-a-time. Therefore, initially, a constraint handling technique was formulated to allow particles to update their best historical solutions according to the feasibility. Results of the sensitivity analysis revealed that PSO was most sensitive to the inertia weight, cognitive component, and social component. The optimal parameter set, determined from the sensitivity analysis, was verified by comparison with metaheuristic methods. The verification study shows that the proposed parameter setting outperformed the other methods in all but one case, where it performed competently.

1. Introduction

Particle Swarm Optimization (PSO) method belongs to the class of Nature-Inspired Algorithms (NIAs) and takes its inspiration from the foraging behavior seen in a school of fish or a flock of birds [1]. Electrical system distribution [2], multi-robot cooperation [3], public transit route network [4], and flyrock prediction [5] are some examples of the real-life application of PSO. According to the no-free lunch theorem, there does not exist an optimization method that outperforms all other methods across every problem [6] or, in other words, certain types of mathematical problems, the computational cost of getting a solution of certain types of optimization problems, averaged over all problems in the class, is the same for any solution method. This implicitly necessitates the need to optimize a particular class of solution methods in a given area of application. In the literature, research comparing the performance of PSO with other metaheuristic methods display the former's efficiency in obtaining the global minimum with fewer iterations [7–9] and coupled with PSO's use of memory and ease of implementation [10], it is considered one of the most favourable NIA. Nevertheless, it is well documented that PSO suffers from a condition, known as premature convergence, that causes it to fail in obtaining the global minimum and thus converge to a local solution [11]. Therefore, many works have been performed to alleviate this problem by either developing novel modifications or incorporating different mechanisms into PSO. The earliest modification, by Shi and Eberhart [12], introduced the inertia weight which controlled the influence of a particle's previous momentum to its future trajectory. Following a similar trend, Clerc and Kennedy [13] proposed the constriction coefficient to balance the local and global search trade-off by damping the three components of a particle's velocity. These two modifications represent novel mechanisms introduced into PSO to improve the swarm's ability to navigate around the design space and avoid stagnation. Swarm diversity is heavily linked with premature convergence - particles navigating to the same region increases the likelihood of stagnation since there is a lack of diverse information being shared [14]. Chaotic mapping has been introduced into PSO in different forms to provide particles with a certain degree of randomness to help promote diversity and enhance search [15–17]. In HPSO-TVAC [18], a time-varying acceleration coefficient was applied to PSO, to orient particles from a

* Corresponding author.
E-mail addresses: mewael.d@alumni.ubc.ca (M. Isiet), mohamed.gadala@adu.ac.ae, gadala@mech.ubc.ca (M. Gadala).

https://doi.org/10.1016/j.jocs.2020.101086
1877-7503/

global to a local search, with a reinitialization velocity scheme to prevent particle stagnation. In another approach, Zhan et al. [19] split the search into four states – exploration, exploitation, convergence and jumping out, PSO was adapted using the Euclidean distance of each particle as the feedback parameter. Chen et al. [20] promoted diversity by employing a lifespan to the swam leader and allowing particles to challenge for leadership. PSO has also been successfully hybridized with other global optimizers such as the Ant Colony Optimization (PSOACO) [21], Genetic Algorithm (HGAPSO) [22], and Artificial Neural Network (PSO-ANN) [23]. In PSOACO, the global exploration is governed by PSO while ACO controls the localized search where virtual ants use pheromone trails to refine particles' positions. HGAPSO creates a population of individuals by mutation, crossover, and elitism, and this hybridization imitates the maturation phenomenon. In PSO-ANN, motivated to improve the performance of ANN, PSO was employed to determine the optimal parameters of ANN to improve its learning ability. Poor configuration of the control settings in PSO may lead to premature convergence, poor performance, or slow convergence [24,25]. Beielstein et al. [26] conducted an analysis of PSO using design of experiments, this method reduces the number of simulation-runs but does not allow the study of individual parameters, and therefore a conclusive hypothesis cannot be established. A simple sensitivity measure approach known as the one at a time approach is applied in this research work to study the effect produced in the output by changing the value of a single parameter [27,28]. The sensitivity measure is obtained by increasing a parameter with a given increment while leaving the other parameters fixed and then quantifying the output change in the model. This approach allows the user to know which parameter is responsible for the failure of the model.

Many real-world engineering design problems are formulated as constrained optimization problems. Presence of constraints has a significant effect on the performance of an optimization algorithm. The original form of any NIA lacks a system for handling constraints [29]. Constraint handling techniques are required to provide necessary information pertaining to the feasibility of a specific solution [30]. There exist many types such as the static penalty [31], dynamic penalty [32], death penalty [33], and the feasibility-based penalty [34]. In this work, the control parameters in PSO are studied by restricting particles to feasible regions. The organization of this work is as follows: Chapter 2 introduces the PSO algorithm, its control parameters, describes some developed strategies and the constraint handling approach. Chapter 3 illustrates the constraint handling technique applied to PSO. Chapter 4 introduces the sensitivity analysis study and numerical results. Finally, Chapter 5 concludes the research work.

2. Particle Swarm Optimization (PSO) overview

In PSO [1], each particle represents a design point in the hyperspace, and its current location is denoted by $\boldsymbol{x}^{(i,k)}$ where i is the individual particle and k is the iteration number. Each particle keeps track of its current position and its best solution so far denoted by $\boldsymbol{x}_P^{(i,k)}$. Another value that is tracked by the algorithm is the best global solution denoted by $\boldsymbol{x}_G^{(k)}$. The particle velocity $\boldsymbol{v}^{(i,k)}$ refers to the change in particle location at each iteration k [35]. In short, the PSO algorithm attempts to accelerate particles by updating its velocity towards its own personal best and the global best solution. Accordingly, $\boldsymbol{v}^{(i,k)}$ and $\boldsymbol{x}^{(i,k)}$ are both iteratively updated by the following two equations respectively:

$$\boldsymbol{v}^{(i,k+1)} = \boldsymbol{v}^{(i,k)} + c_1 \boldsymbol{rand}_1{}^{(k)} \left(\boldsymbol{x}_P^{(i,k)} - \boldsymbol{x}^{(i,k)}\right) + c_2 \boldsymbol{rand}_2{}^{(k)} \left(\boldsymbol{x}_G^{(k)} - \boldsymbol{x}^{(i,k)}\right);\ i = 1\ to\ N \tag{1}$$

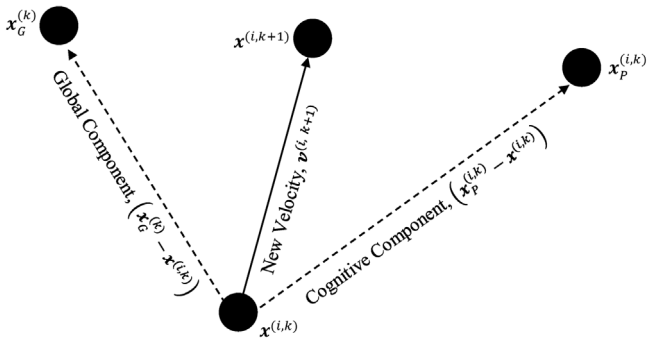

Fig. 1. Particle velocity and position update in a two-dimensional search space.

$$\boldsymbol{x}^{(i,k+1)} = \boldsymbol{x}^{(i,k)} + \boldsymbol{v}^{(i,k+1)};\ i = 1\ to\ N \tag{2}$$

where c_1 and c_2, the cognitive and social parameter respectively, are collectively known as the acceleration coefficients. $\boldsymbol{rand}_1{}^{(k)}$ and $\boldsymbol{rand}_2{}^{(k)}$ are vectors containing random numbers generated at every k iteration ($0 \le rand \le 1$), and N refers to the swarm size. In the particle velocity-update equation, three vector components exist; namely the momentum vector, the cognitive or personal component and the social or global component. Fig. 1 illustrates how the three velocity vector components influence a particle towards the search for the global best solution. The above two equations represent the original PSO algorithm. Once the particles' new velocity and position have been calculated, $\boldsymbol{x}_P^{(i,k)}$ and $\boldsymbol{x}_G^{(k)}$ are updated iteratively with the following equations respectively:

$$\boldsymbol{x}_P^{(i,k)} = \begin{cases} \boldsymbol{x}^{(i,k)}, & \text{if} f\left(\boldsymbol{x}^{(i,k)}\right) < f\left(\boldsymbol{x}_P^{(i,k)}\right) \\ \boldsymbol{x}_P^{(i,k)}, & \text{if} f\left(\boldsymbol{x}^{(i,k)}\right) \ge f\left(\boldsymbol{x}_P^{(i,k)}\right) \end{cases} \tag{3}$$

$$\boldsymbol{x}_G^{(k)} = \begin{cases} \boldsymbol{x}^{(i,k)}, & \text{if} f\left(\boldsymbol{x}^{(i,k)}\right) < f\left(\boldsymbol{x}_G^{(k)}\right) \\ \boldsymbol{x}_P^{(i,k)}, & \text{if} f\left(\boldsymbol{x}^{(i,k)}\right) \ge f\left(\boldsymbol{x}_G^{(k)}\right) \end{cases} \tag{4}$$

Both the particle's position and velocity are bounded according to a predefined limit. The dynamic range of a particle's position is defined by the design variable limit, whereas the velocity clamping-limit determines the particle's velocity range. In terms of initialization, the particles' positions are randomly distributed across the hyperspace, while their velocities are set as zero [36]. The initial diversity of the swarm profoundly influences the efficiency of the algorithm since the higher the diversity, the higher the chances of them locating the global solution are [37]. Eqs. (1–4) are updated at every iteration until a predefined termination criterion is met. Typically, the algorithm terminates once an acceptable global solution is obtained or a predefined maximum number of function evaluations or iterations is reached [38].

2.1. Control parameters

The original PSO version described in the previous section has certain parameters, known as the control parameters, which have an impact on the overall search capabilities of the algorithm. These control parameters, namely the acceleration coefficients, velocity clamping-limit, swarm size and maximum number of iterations, are briefly described as:

- **Acceleration coefficient**: The terms c_1 and c_2 when changed could lead to particles flying to undesirable regions, therefore the acceleration coefficients limits have both been set as 2 since the conception of the PSO method [1]. However, this could be harmful

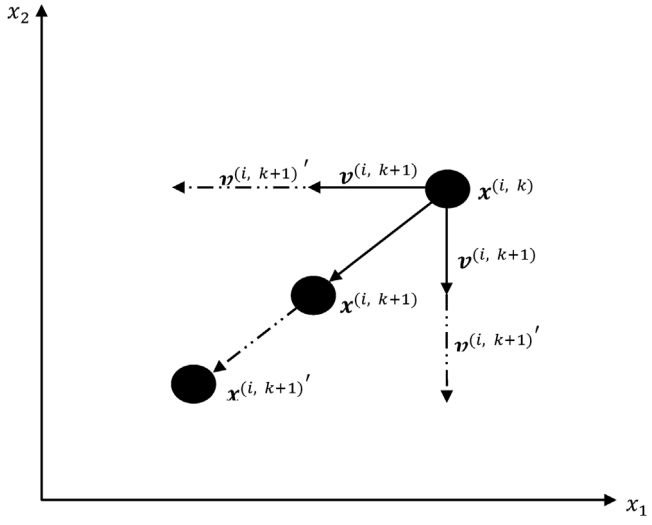

Fig. 2. Effect of Clamping-Limit in Updating the Position of Particles.

to the overall search for the optimal solution since these parameters are key in controlling the effectiveness and efficiency of the algorithm [39]. Therefore, a time varying acceleration coefficient was developed to guide the particles to a global search ($c_1 > c_2$) in the initial stages and then gradually direct them to a localized search ($c_2 > c_1$) in the final iterations [18]. The acceleration coefficients time varying forms are presented below, where ${c_1}^k$ and ${c_2}^k$ are the cognitive and social parameters respectively at iteration, c_{1max} and c_{2max} are the maximum cognitive and social parameter respectively, c_{1min} and c_{2min} are the minimum cognitive and social parameter respectively, k is the current iteration and Max_k is the maximum iteration.

$$ {c_1}^k = \left(\frac{(c_{1max} - c_{1min}) \times (Max_k - k)}{Max_k} \right) + c_{1min} \tag{5} $$

$$ {c_2}^k = c_{2max} - \left(\frac{(c_{2max} - c_{2min}) \times (Max_k - k)}{Max_k} \right) \tag{6} $$

- **Velocity Clamping-limit**: In order to clamp the maximum step size change, the particles' velocities are bounded by $(-\boldsymbol{V}_{max}\ \boldsymbol{V}_{max})$ [40]:

$$ \boldsymbol{V}_{max} = \delta \times (\boldsymbol{x}_{max} - \boldsymbol{x}_{min}) \tag{7} $$

where δ is a clamping constant set between (0.1∼1.0) and $\boldsymbol{x}_{max}$ and $\boldsymbol{x}_{min}$ are the maximum and minimum value of the design variables [41]. The value of the maximum clamping-limit controls the coarseness of the search by restricting the step size (velocity) changes [24]. When a large V_{max} is set, global exploration is promoted, however a very large value could lead to particles overflying the global optimum leading to premature convergence. On the other hand, a smaller V_{max} facilitates local search but a very small value would lead to particles requiring more iterations to solve the problem [42,43]. The effect of particle velocity clamping-limit for a two-dimensional design space is illustrated in Fig. 2. In this figure, $\boldsymbol{x}^{(i,k+1)'}$ represents the position of particle i without the effect of the clamping limit. The position $\boldsymbol{x}^{(i,k+1)}$ is the result of the clamping-limit on both dimensions, x_1 and x_2. The velocity $\boldsymbol{v}^{(i,\ k+1)'}$ is the velocity of particle i without the clamping limit and $\boldsymbol{v}^{(i,\ k+1)}$ with clamping-limit.

- **Swarm Size**: Swarm size, N, plays an important role in the performance of the algorithm. This property refers to the number of particles in the hyperspace – it explicitly affects the diversity in the swarm. When a large swarm size is selected, the particles cover more space in the search area but at the expense of computational efforts. According to Arora [44], anywhere between 5 $x_N \sim 10\, x_N$ (where x_N is the number of design variables) is a good estimate, however in some other works anywhere between 10 ∼ 30 particles [45–47] proved to be successful in solving common optimization benchmark problems.
- **Maximum number of iterations**: Maximum number of iterations, Max_k, refers to the number of iterations the PSO algorithm goes through until it terminates. If the maximum number of iterations is set as the termination criteria, then the predefined value has a direct impact on the likelihood of the algorithm discovering the global solution. Therefore, when a smaller value is selected, the probability of PSO obtaining the global solution is lower than if a larger value was selected. However, the downside of a large maximum number of iterations is the increased computational cost [42]. Therefore, the phenomena of premature convergence could occur due to the poor selection of a maximum number of iterations.

2.2. Inertia weight strategies

The introduction of the inertia weight helped promote better searching from particles by balancing the global search and the local search [12]. The inertia weight was incorporated into Eq. (1) as follows:

$$ \boldsymbol{v}^{(i,\ k+1)} = w\boldsymbol{v}^{(i,\ k)} + c_1 \boldsymbol{rand}_1{}^{(k)} \left(\boldsymbol{x}_P^{(i,k)} - \boldsymbol{x}^{(i,k)} \right) + c_2 \boldsymbol{rand}_2{}^{(k)} \left(\boldsymbol{x}_G^{(k)} - \boldsymbol{x}^{(i,k)} \right) \tag{8} $$

where w is an inertia damping term that influences the momentum vector. A large value enhances the particle's global search, and a small value increases the particle's local search. Shi and Eberhardt [12] demonstrated a significant improvement in the performance of PSO when the inertia weight linearly varied between 0.9 to 1.2. In a subsequent study on the linearly decreasing inertia weight (PSO-LDIW), Eberhardt and Shi [48] reported that varying the inertia weight between 0.9 to 0.4 resulted in improved performance. The linearly decreasing inertia weight equation is described as [42]:

$$ w^{(k)} = \left(\frac{(w_{max} - w_{min}) \times (Max_k - k)}{Max_k} \right) + w_{min} \tag{9} $$

where w_{min} is the minimum inertia weight value, w_{max} is the maximum inertia weight, k is the current iteration, and Max_k is the maximum iteration. $w^{(k)}$ decreases from w_{max} to w_{min} with respect to k and Max_k. Naturally, different dynamic strategies could be applied to the inertia weight. For example, the random inertia weight (PSO-RIW) by Shi and Eberhart [49] randomly determines the inertia weight at a given iteration according to:

$$ \left(w^k = 0.5 + \frac{rand^k}{2} \right) \tag{10} $$

where $rand^k$ is a random number [0, 1], and the range of w^k is between [0.5, 1]. It was reported that RIW improved the robustness of the PSO algorithm [49]. Another effective strategy was the use of a fuzzy system to adapt the inertia weight, which proved to be successful in solving dynamic optimization problems [50]. Some researchers utilized previously developed versions of the inertia weight to enhance its influence in the particles' search capability. By introducing an exponential component into the PSO-LDIW,

the nonlinear inertia weight (PSO-NLIW) [51] was devised with the following equation:

$$w^{(k)} = \left(\frac{(w_{max} - w_{min}) \times (Max_k - k)^c}{(Max_k)^c} \right) + w_{min} \tag{11}$$

where c is the nonlinear modulation index. It was reported that when $c = 0.9 \sim 1.3$ the inertia weight managed to efficiently navigate the particles between a globalized search in the early iterations to a more refined search in the later iterations [51]. In another adaption, a chaotic inertia weight (PSO-CIW) was proposed by introducing a chaotic mechanism into the PSO-LDIW [16] as follows:

$$w^{(k)} = \left(\frac{(w_{max} - w_{min}) \times (Max_k - k)}{(Max_k)} + w_{min} \right) \times z^k$$
$$z^k = \alpha \times z^{k-1} \times \left(1 - z^{k-1}\right) \tag{12}$$

where z is the chaotic term and when $3.75 \leq \alpha \leq 4$ the inertia weight behaves chaotically. Moreover, when $\alpha = 4$, the chaotic term covers an interval of [0,1] if $z^0 \in [0, 1]$ and $z^0 \notin [0, 0.25, 0.5, 0.75, 1.0]$. This strategy promotes more randomness and thus increases the swarm's diversity [52]. Furthermore, empirical results demonstrated that the chaotic inertia weight managed to improve PSO's convergence accuracy, convergence speed, and exploration search ability [16].

2.3. Constraint handling approach

PSO modifications are needed to handle constrained problems [33]. By the preservation of feasible solutions, particles can search the whole design space but keep track of the possible solutions only, and this can be done through the tracking of constraint violations. In the current approach, the following logical conditions are applied for handling constraints:

- Condition 1: Any feasible solution is chosen over an infeasible solution.
- Condition 2: Between two feasible solutions, the one with the better objective function value is preferred.
- Condition 3: Between two infeasible solutions, the one with the better objective function value is preferred.

By applying the first condition, the algorithm orients itself to favor feasible solutions over infeasible ones, whereas using the second condition forces the algorithm to direct the search towards the feasible regions with better solutions [53]. Fig. 3 shows a schematic of the applied constraint handling technique in UAPSO. As can be seen in Fig. 3(a), $f\left(\boldsymbol{x}^{(i,k+1)}\right) < f\left(\boldsymbol{x}^{(i,k)}\right)$ however $\boldsymbol{x}^{(i,k+1)}$ is not a feasible solution therefore $\boldsymbol{x}^{(i,k)}$ is retained by the algorithm as the best solution. In Fig. 3(b) since $\boldsymbol{x}^{(i,k+1)}$ is a feasible solution and $\left(\boldsymbol{x}^{(i,k+1)}\right) < f\left(\boldsymbol{x}^{(i,k)}\right)$, $\boldsymbol{x}^{(i,k+1)}$ is retained by the algorithm. In case both $\boldsymbol{x}^{(i,k)}$ and $\boldsymbol{x}^{(i,k+1)}$ are infeasible and $\left(\boldsymbol{x}^{(i,k+1)}\right) < f\left(\boldsymbol{x}^{(i,k)}\right)$, $\boldsymbol{x}^{(i,k+1)}$ is retained by the algorithm as the best solution, as shown in Fig. 3(c).Reinitialization of particles' initial position until all are within the feasible region was suggested as a method [33] to avoid premature convergence in an infeasible region. However, this approach may increase the computational time. By applying a penalty on unfeasible solutions [54], PSO avoids premature convergence when coupled with the abovementioned conditions. Two pieces of information are included in the penalty function [55] – the number of constraint violations and the amount by which the constraint was violated. The modified PSO algorithm solves the cost function as such:

$$F_i(\boldsymbol{x}) = \begin{cases} f_i(\boldsymbol{x}), & \text{if feasible solution} \\ f_i(\boldsymbol{x}) + \beta_1 \left(\sum_{i=1}^{d} h \right) + \beta_2 \left(\sum_{i=1}^{d} y \right), & \text{if infeasible solution} \end{cases} \tag{13}$$

where $F_i(x)$ is the penalty function, $f_i(x)$ is the original cost function, $\sum_{i=1}^{d} h$ is the sum of d violated constraints, $\sum_{i=1}^{d} y$ is the sum of the amount of d violated constraints and β_1 and β_2 are the penalty factors. Both β_1 and β_2 are set to the value of 10^3.

3. PSO sensitivity analysis

The speed-reducer optimization problem [56] is a common benchmark example used for testing and comparing optimization methods and is displayed in Fig. 4. Here it is considered for the sensitivity analysis. In this problem, the objective is to minimize the weight of the speed-reducer while optimizing seven design variables – face width (b), module of teeth (m), number of teeth in the pinion (z), length of the first shaft between bearings (l_1), length of the second shaft between bearings (l_2), and the diameter of first (d_1) and second shafts (d_2), respectively. All design variables are continuous apart from z which is an integer value and handled using the truncated method [57]. Furthermore, the problem is constrained by 11 constraints. The problem description is given in Appendix (A). The global minimum of this problem is 2.994 E + 04 [58].

3.1. Numerical experimental setup

The sensitivity analysis included the five control parameters of PSO. The analysis determined the sensitivity of the performance of PSO to variations in each parameter using a statistical approach. The speed-reducer problem is solved by PSO 100 times for each parameter set. The best global solution is stored at the end of each run and ranked after 100 runs to determine the best and worst solutions obtained. The mean value and standard deviation are calculated by the following equations respectively:

$$f(x_M) = \frac{\sum_{i=1}^{100} f\left(x_G^i\right)}{100} \tag{14}$$

$$\mathrm{SD} = \sqrt{\frac{\sum_{i=1}^{100} \left(f\left(x_G^i\right) - f(x_M)\right)^2}{100 - 1}} \tag{15}$$

where $f(\boldsymbol{x}_M)$ is the mean value obtained across 100 runs, $f\left(\boldsymbol{x}_G^i\right)$ is the best global solution obtained at run i and SD is the standard deviation. The sensitivity analysis was formulated using the numerical computing software MATLAB (R2017a – Academic Version) and the simulations were run on Intel Core i7 with 16 GB RAM.

3.2. Sensitivity analysis result – maximum number of iterations (Max_k)

Max_k depends on the nature of the problem. A low value decreases the probability of obtaining the global solution, whereas a high value increases the computational demands. Max_k is studied within the range of [100, 1000], which was used by Danial et al. in their study [23]. Guerdia [59] recommended a range between 200–400 iterations for highly complex problems and between 10–50 iterations for medium complex problems. For this sensitivity analysis, the range used by Danial et al. is selected since it will help in detecting the sensitivity of PSO with Max_k as it covers a broad

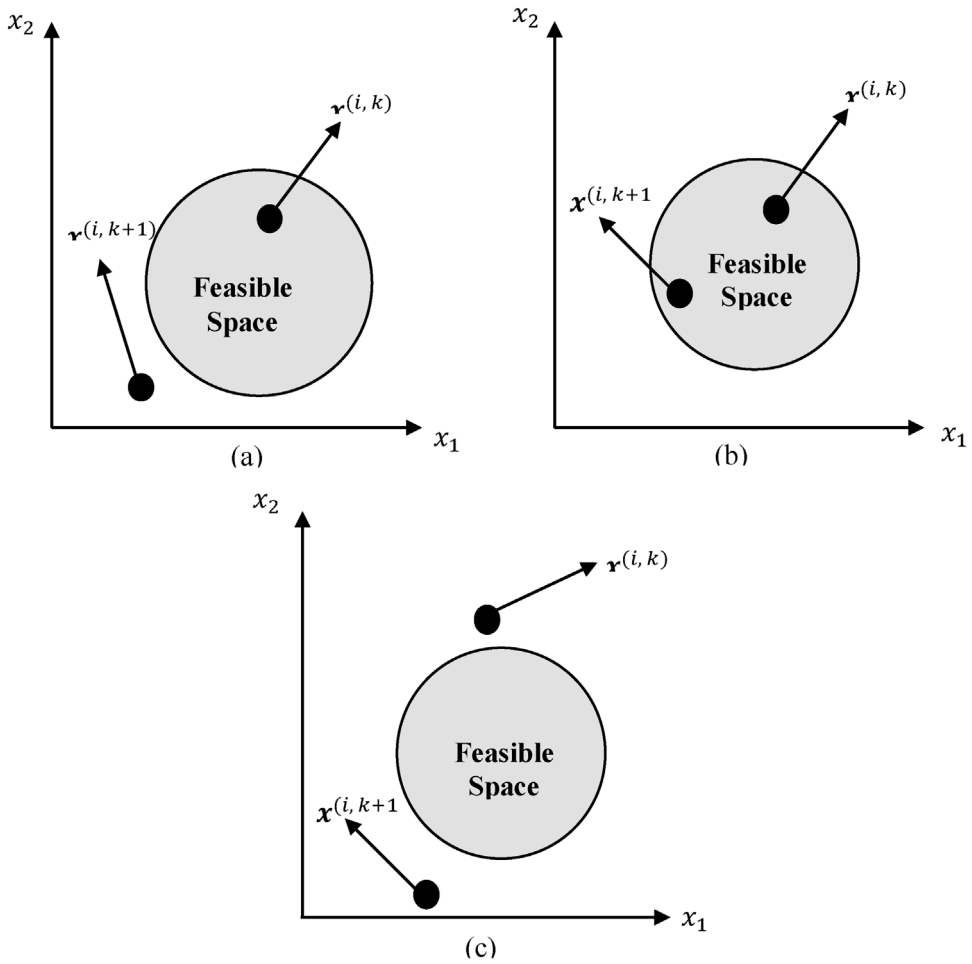

Fig. 3. Schematic diagram of the constraint handling approach with (a) showing preference towards feasibility, (b) showing preference towards feasible solutions with better fitness and (c) showing preference towards infeasible solutions with better fitness.

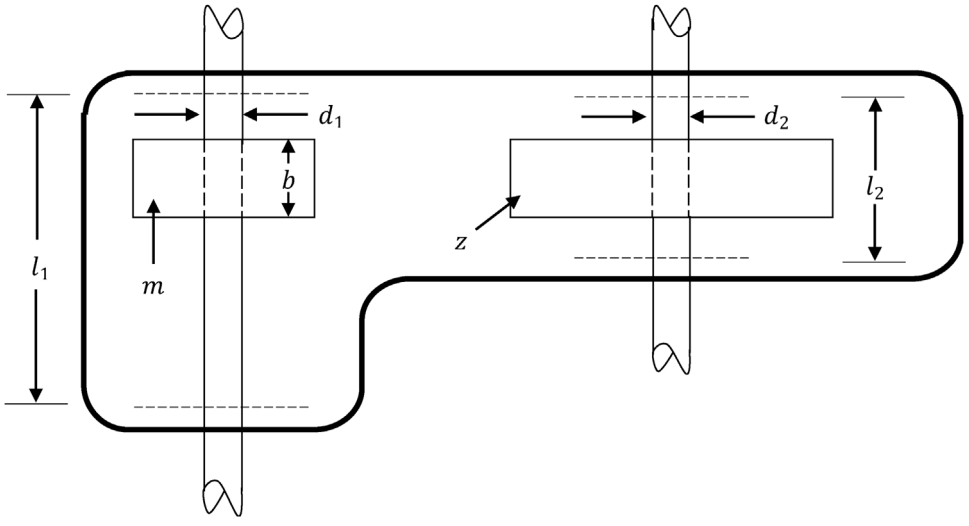

Fig. 4. Schematic representation of the speed-reducer design problem.

range. As mentioned earlier, Max_k has a significant influence on the probability of the swarm locating the global minimum, a small Max_k would not provide particles enough time to explore the design space and a large Max_k would increase computational demand. In this analysis, the performance of PSO with varying Max_k is studied with 50 particles, a value of 2 for each acceleration coefficient, a velocity clamping constant of 1 and the inertia weight is set as 0.5. The results of the sensitivity analysis can be seen in Table 1. Increasing the number of iterations led to a change in the performance of PSO up to a certain point, as evidenced by a homogenous statistical trend from Max_k 200 onwards; this shows that increasing computational effort can be superfluous at times. Mean performance shows

Table 1
Sensitivity Analysis of Max_k with $N = 50$, $\delta = 1$, $c_1 = c_2 = 2$, and $w = 0.5$.

Max_k	Best	Worst	Mean (Standard Deviation)
100	2.994 E + 03	3.056 E + 03	3.016 E + 03 (2.032 E + 01)
200	2.994 E + 03	3.047 E + 03	3.014 E + 03 (2.035 E + 01)
300	2.994 E + 03	3.047 E + 03	3.013 E + 03 (2.050 E + 01)
400	2.994 E + 03	3.047 E + 03	3.013 E + 03 (1.962 E + 01)
500	2.994 E + 03	3.047 E + 03	3.012 E + 03 (1.957 E + 01)
600	2.994 E + 03	3.047 E + 03	3.013 E + 03 (2.004 E + 01)
700	2.994 E + 03	3.047 E + 03	3.013 E + 03 (1.985 E + 01)
800	2.994 E + 03	3.047 E + 03	3.012 E + 03 (1.957 E + 01)
900	2.994 E + 03	3.047 E + 03	3.013 E + 03 (2.004 E + 01)
1000	2.994 E + 03	3.047 E + 03	3.013 E + 03 (1.828 E + 01)

Table 2
Sensitivity Analysis of N with $Max_k = 500$, $\delta = 1$, $c_1 = c_2 = 2$, and $w = 0.5$.

N	Best	Worst	Mean (Standard Deviation)
5	2.994 E + 03	3.357 E + 03	3.083 E + 03 (8.679 E + 01)
25	2.994 E + 03	3.141 E + 03	3.026 E + 03 (2.628 E + 01)
50	2.994 E + 03	3.047 E + 03	3.012 E + 03 (1.957 E + 01)
100	2.994 E + 03	3.043 E + 03	3.002 E + 03 (1.223 E + 01)
150	2.994 E + 03	3.017 E + 03	2.997 E + 03 (4.950 E + 00)
200	2.994 E + 03	3.007 E + 03	2.995 E + 03 (2.822 E + 00)
250	2.994 E + 03	3.007 E + 03	2.994 E + 03 (2.690 E + 00)
300	2.994 E + 03	2.994 E + 03	2.994 E + 03 (3.665 E-12)
350	2.994 E + 03	2.994 E + 03	2.994 E + 03 (2.825 E-12)
400	2.994 E + 03	2.994 E + 03	2.994 E + 03 (1.828 E-12)
450	2.994 E + 03	2.994 E + 03	2.994 E + 03 (1.828 E-12)
500	2.994 E + 03	2.994 E + 03	2.994 E + 03 (1.828 E-12)

that increasing Max_k doesn't result in a more efficient performance, a saturation point is reached where larger iterations do not impact the performance of PSO. This study also reveals the importance of the other control parameters in navigating the particles for avoiding premature convergence. Max_k can improve performance, but cannot improve PSO's efficiency and robustness since it only controls the duration of search and not the navigation of particles in the design space.

3.3. Sensitivity analysis result – swarm size (N)

Shi and Eberhardt [60] claimed that swarm sizes have minimal effect in the performance of PSO by studying the impact of the swarm size within a range of [20, 160], Daniel et al. [23] employed a range between 25 and 500 particles and Arora [44] suggested that the swarm size should be selected based on the number of variables. Since there does not exist a mechanism for the determination of the swarm size a priori, a parametric study is typically conducted [61]. A sensitivity analysis of the swarm size is conducted for a range of 5–500 particles, a value of 2 for each acceleration coefficient, a velocity clamping constant of 1, the inertia weight is set as 0.5 and with an iteration of 500. As expected, the results display that by increasing diversity, PSO performs more efficiently and effectively; however, this is at the expense of computational effort. In Table 2, it can be seen that between a population of 300 and 500 similar statistical performance was recorded. Increasing the swarm size, increases the likelihood of particles locating the global minimum but exceeding a certain threshold leads to a homogenous performance. Even though small populations managed to obtain the global minimum, the navigation of swarms with different sizes changes. Fig. 5 displays the historical performance of PSO with varying swarm sizes. Population sizes of 250, and 500 perform very similar, as seen in Fig. 5, and this agrees with their respective statistical performance. However, the performance of larger swarm sizes (250 and 500) differ from the smaller sizes (5 and 50), which disagrees with Shi and Eberhardt's [60] results. However, this is mostly likely due to the constrained environment, where particles

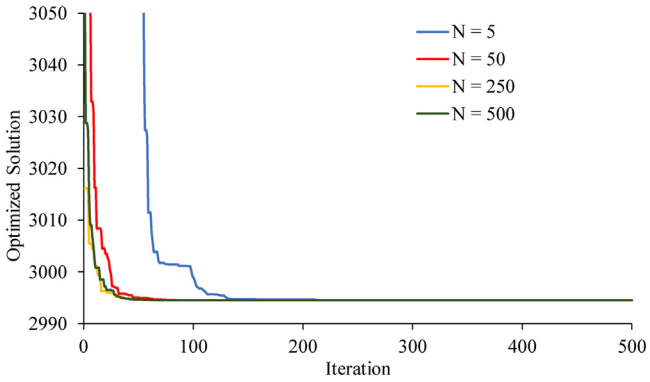

Fig. 5. Effect of swarm size on the performance of PSO.

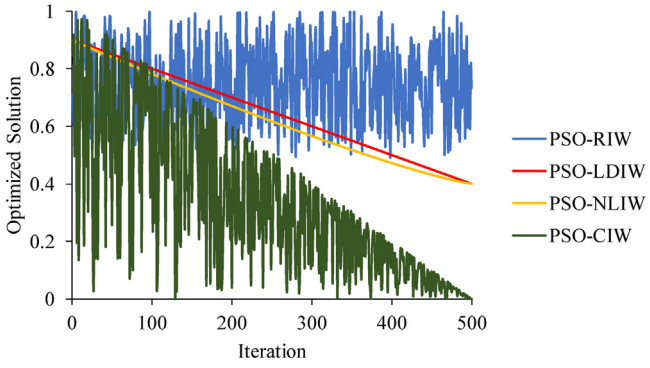

Fig. 6. Dynamic behaviour of the considered inertia weight strategies.

Table 3
Range of the considered dynamic inertia weights.

PSO Variant	PSO-LDIW [12]	PSO-NLIW [51]	PSO-RIW [49]	PSO-CIW [16]
w Range	0.4 - 0.9	0.4 - 0.9	0.5 - 1.0	0.0 - 1.0

behave in a different manner since particles update their positions according to the feasibility of their performance.

3.4. Sensitivity analysis result – inertia weight ($\boldsymbol{w}$)

Inertia weight dictates the influence of a particle's previous momentum in updating its velocity which enables more control over the trajectory of a particle, i.e. exploration or exploitation. A range of constant inertia weights was applied by Shi and Eberhardt [12], and they concluded that PSO performs best with a moderate number of iterations when an inertia weight between the range of [.8, 1.2] is employed. Bansal et al. [62] studied the performance of PSO with different types of dynamic inertia weights and a constant inertia weight ($w = 0.7$). A momentum-less ($w = 0$) PSO was also studied by Shi and Eberhardt [12] where it was found that premature convergence occurred in particles close to the global best solution due to the lack of momentum. In the conception of PSO [1], the inertia weight took a value of unity ($w = 1.0$). PSO-LDIW, PSO-RIW, PSO-NLIW, and PSO-CIW are compared to different constant inertia weights. Based on the abovementioned work, a range of constant inertia weight [0, 1.2] weight is implemented. The dynamic behavior of the considered inertia weights is displayed in Fig. 6, and illustrates the increased randomness introduced to the particles by PSO-RIW and PSO-CIW. The ranges of the considered dynamic inertia weights are given in Table 3. Increasing the inertia weight resulted in a decrease in the statistical performance of PSO. The constant inertia weight between [0, 0.5] managed to outperform the dynamic variants of the inertia weight, which is

Table 4a
Sensitivity Analysis of w with Max_k = 500, δ = 1, $c_1 = c_2$ = 2, and N = 50.

w	Best	Worst	Mean (Standard Deviation)
0.0	2.994 E + 03	2.994 E + 03	2.995 E + 03 (2.964 E + 00)
0.1	2.994 E + 03	3.034 E + 03	2.999 E + 03 (1.104 E + 01)
0.2	2.994 E + 03	3.034 E + 03	3.003 E + 03 (1.428 E + 01)
0.3	2.994 E + 03	3.047 E + 03	3.009 E + 03 (1.797 E + 01)
0.4	2.994 E + 03	3.047 E + 03	3.010 E + 03 (1.802 E + 01)
0.5	2.994 E + 03	3.047 E + 03	3.012 E + 03 (1.957 E + 01)
0.6	2.994 E + 03	3.056 E + 03	3.018 E + 03 (2.047 E + 01)
0.7	2.994 E + 03	3.056 E + 03	3.023 E + 03 (2.048 E + 01)
0.8	2.994 E + 03	3.056 E + 03	3.029 E + 03 (1.997 E + 01)
0.9	2.995 E + 03	3.150 E + 03	3.037 E + 03 (2.197 E + 01)
1.0	3.012 E + 03	3.150 E + 03	3.051 E + 03 (1.807 E + 01)
1.1	3.014 E + 03	3.183 E + 03	3.060 E + 03 (3.038 E + 01)
1.2	3.037 E + 03	3.198 E + 03	3.062 E + 03 (3.366 E + 01)
PSO-LDIW	2.994 E + 03	3.056 E + 03	3.030 E + 03 (1.897 E + 01)
PSO-RIW	2.994 E + 03	3.056 E + 03	3.025 E + 03 (2.084 E + 01)
PSO-NLIW	2.994 E + 03	3.056 E + 03	3.029 E + 03 (2.090 E + 01)
PSO-CIW	2.994 E + 03	3.047 E + 03	3.017 E + 03 (2.009 E + 01)

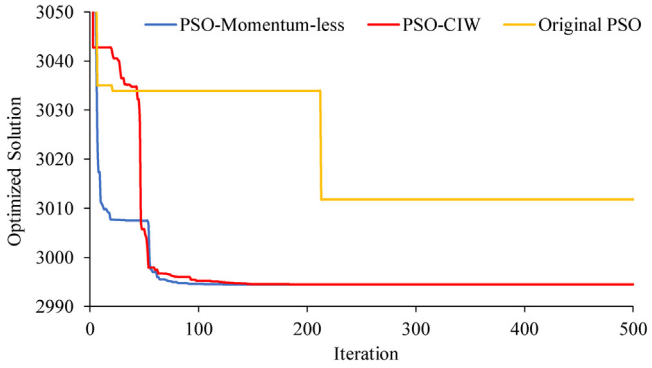

Fig. 7. Comparison of the convergence of PSO-Momentum-less, PSO-CIW and the original PSO.

in contrary to multiple studies [12,62], as seen in Tables 4a and 4b. Larger inertia weight (0.9–1.2) failed to obtain the global minimum, and performed inefficiently. As for the performance of the dynamic inertia weights, PSO-CIW managed to perform best due to the increased randomness introduced to the particles, thereby promoting diversity in the search leading to a more robust search. In Fig. 7, the convergence curves of the optimal solution display that the momentum-less version managed to converge to the global minimum with fewer iterations and thus performed more efficiently than the CIW version, while the original PSO version prematurely converged to a local solution.

3.5. *Sensitivity analysis result – acceleration coefficients (c_1 and c_2)*

In a study [63] of the behavior of the particles in PSO, Kennedy suggested that the sum value of both the cognitive and social component in the particle velocity-update equation should not exceed 4 and this has been established from the conception of PSO [1]. Ozcan and Mohan [64] demonstrated through theoretical analysis that particles exhibit high levels of oscillations when the sum of the acceleration coefficients exceeds 4. Kennedy [65] studied the performance of a social-only ($c_1 = 0$ and $c_2 = 4$) and a cognitive-only PSO ($c_1 = 4$ and $c_2 = 0$) where he demonstrated that the cognitive-only PSO had a quicker convergence and efficiency. Clerc [66] suggested the use of a constriction factor (equivalent to $w = 0.729$ and $c_1 = c_2 = 1.49445$) to ensure convergence, whereas Trelea [46] demonstrated that $w = 0.6$ and $c_1 = c_2 = 1.7$ resulted in a good performance. After running PSO with different combinations of acceleration coefficients, Danial et al. [23] displayed the superiority of $c_1 = 2.286$ and $c_2 = 1.714$. Van den Bergh and Engelbrecht [24] conducted a theoretical analysis of particle trajectories and convergence and developed a condition, based on the inertia weight and acceleration coefficients, that guarantees particle convergence. As such, the sensitivity analysis of the acceleration coefficient is studied with the studies mentioned above and further generated combinations that satisfy the conclusions of Kennedy [63], Ozcan and Mohan [64] and Van den Bergh and Engelbrecht [24]. Furthermore, the time-varying acceleration coefficients, implemented by Ratnaweera et al. [18], is also included to examine whether the promotion of global exploration in the early iterations and local exploitation in the latter stages can effectively navigate the swarm. Also, the nonlinear version of TVAC (NL-TVAC) developed by Yu et al. [67] is considered and described as follows:

$$c_1^k = c_{11} + \sin\left(\frac{k \times \pi}{2 \times Max_k}\right) \tag{16}$$

$$c_2^k = c_{22} - \sin\left(\frac{k \times \pi}{2 \times Max_k}\right) \tag{17}$$

where $c_{11} = 1$ and $c_{22} = 2$. In TVAC [18], c_1^k varies from 2.5 to 0.5 and c_2^k from 0.5 to 2.5. It should be noted that apart from the parameter selection of Clerc [66] and Trelea [46], the inertia weight was set to 0.5. An iteration of 500, velocity clamping constant of 1 and swarm size of 50 was employed. The results, displayed in

Table 4b
Sensitivity Analysis of c_1 and c_2 with Max_k = 500, δ = 1, and N = 50.

c_1	c_2	$c_1 + c_2$	Best	Worst	Mean (Standard Deviation)
3.5	0.25	3.75	3.013 E + 03	3.498 E + 03	3.159 E + 03 (1.094 E + 03)
3.25	0.5	3.75	2.994 E + 03	3.085 E + 03	3.007 E + 03 (1.819 E + 03)
3.0	0.5	3.5	2.994 E + 03	3.078 E + 03	3.001 E + 03 (1.652 E + 03)
2.5	1.0	3.5	2.994 E + 03	3.034 E + 03	2.997 E + 03 (8.978 E + 00)
2.0	1.75	3.75	2.994 E + 03	3.047 E + 03	3.012 E + 03 (1.923 E + 01)
2.0	2.0	4.0	2.994 E + 03	3.047 E + 03	3.012 E + 03 (1.957 E + 01)
1.75	2.0	3.75	2.994 E + 03	3.047 E + 03	3.016 E + 03 (1.941 E + 01)
1.0	2.5	3.5	2.994 E + 03	3.141 E + 03	3.032 E + 03 (2.439 E + 01)
0.5	3.0	3.5	2.994 E + 03	3.222 E + 03	3.063 E + 03 (5.372 E + 01)
0.5	3.25	3.75	2.994 E + 03	3.222 E + 03	3.062 E + 03 (5.825 E + 01)
0.25	3.5	3.75	2.994 E + 03	3.222 E + 03	3.077 E + 03 (6.673 E + 01)
0.0	4.0	4.0	2.994 E + 03	3.222 E + 03	3.104 E + 03 (7.679 E + 01)
1.49445	1.49445	2.9889	2.994 E + 03	3.047 E + 03	3.021 E + 03 (2.070 E + 01)
1.7	1.7	3.4	2.994 E + 03	3.047 E + 03	3.012 E + 03 (2.020 E + 01)
2.286	1.714	4.0	2.994 E + 03	3.047 E + 03	3.009 E + 03 (1.761 E + 01)
1.714	2.286	4.0	2.994 E + 03	3.056 E + 03	3.022 E + 03 (2.195 E + 01)
NL-TVAC		3.0	2.994 E + 03	3.047 E + 03	3.017 E + 03 (2.101 E + 01)
TVAC		3.0	2.994 E + 03	3.034 E + 03	2.998 E + 03 (1.015 E + 01)

Table 5
Sensitivity Analysis of $\boldsymbol{V}_{max}$ with Max_k = 500, w = 1, $c_1 = c_2$ = 2 and N = 50.

δ	Best	Worst	Mean (Standard Deviation)
No Velocity Clamping	2.994 E + 03	3.047 E + 03	3.013 E + 03 (1.962 E + 01)
0.1	2.994 E + 03	3.047 E + 03	3.011 E + 03 (1.974 E + 01)
0.2	2.994 E + 03	3.034 E + 03	3.006 E + 03 (1.694 E + 01)
0.3	2.994 E + 03	3.047 E + 03	3.010 E + 03 (1.938 E + 01)
0.4	2.994 E + 03	3.047 E + 03	3.006 E + 03 (1.702 E + 01)
0.5	2.994 E + 03	3.047 E + 03	3.012 E + 03 (2.006 E + 01)
0.6	2.994 E + 03	3.047 E + 03	3.015 E + 03 (2.002 E + 01)
0.7	2.994 E + 03	3.047 E + 03	3.012 E + 03 (1.899 E + 01)
0.8	2.994 E + 03	3.047 E + 03	3.015 E + 03 (2.049 E + 01)
0.9	2.994 E + 03	3.047 E + 03	3.011 E + 03 (1.796 E + 01)
1.0	2.994 E + 03	3.047 E + 03	3.012 E + 03 (1.957 E + 01)
$\boldsymbol{V}_{max} = \mathbf{X}_{max}$	2.994 E + 03	3.047 E + 03	3.013 E + 03 (1.973 E + 01)
LSE	2.994 E + 03	3.047 E + 03	3.011 E + 03 (1.931 E + 01)
VCLDM	2.994 E + 03	3.047 E + 03	3.016 E + 03 (2.071 E + 01)
RM	2.994 E + 03	3.047 E + 03	3.015 E + 01 (1.948 E + 01)
APS	2.994 E + 03	3.047 E + 03	3.014 E + 01 (1.990 E + 01)

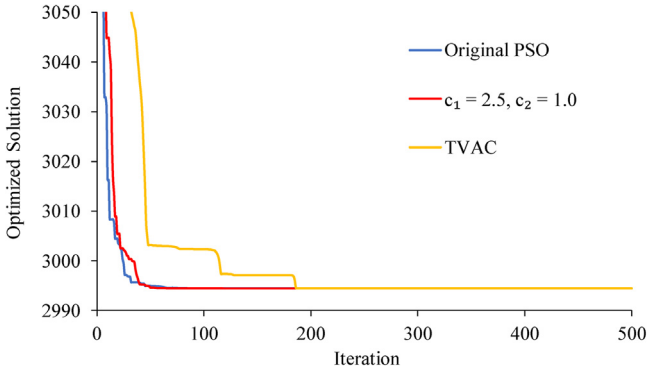

Fig. 8. Comparison of the convergence of PSO ($\boldsymbol{c}_1 = 2.5$, $\boldsymbol{c}_2 = 1.0$), TVAC and the original PSO.

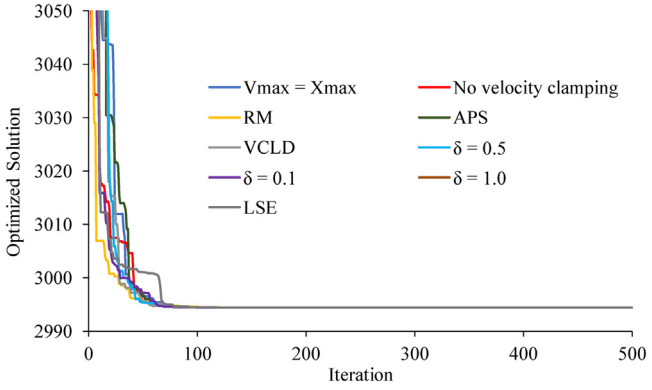

Fig. 9. Comparison of the convergence of PSO ($\boldsymbol{c}_1 = 2.5$, $\boldsymbol{c}_2 = 1.0$), TVAC and the original PSO.

Table 5, show that the conventional setting of the acceleration coefficients did not perform effectively, and this is in agreement with Van den Bergh and Engelbrecht [24]. Due to the absence of a global leader the particles behaved statically in the cognitive-only PSO [65] and thus this version was omitted. The original PSO version managed to perform better than the social-only version; this is in contrary Kennedy [65]. In terms of the dynamic adaptations, TVAC managed to navigate the swarm effectively resulting in a strong performance, and outperformed NL-TVAC. The best performing combination ($c_1 = 2.5$, $c_2 = 1.0$) managed to produce a consistent and efficient performance. In Fig. 8, the convergence curve of the conventional acceleration coefficients is compared with the TVAC and the best performing combination. It can be seen that both constant combinations led to quick convergence whereas TVAC required more iterations as it directed the particles to explore the design space in the initial iterations and then towards the end of the search TVAC promoted convergence to the global solution.

3.6. *Sensitivity analysis result – velocity clamping-limit* (v_{max})

When studying the trajectory of particles, Van den Bergh and Engelbrecht [24] showed that velocity clamping is necessary when employing the original PSO formulation since particles would diverge and leave the desirable regions. Typically, the velocity clamping-limit is set to that of the dynamic range of a particle's position ($\boldsymbol{V}_{max} = \boldsymbol{x}_{max}$) [10,48]. Shahzab et al. [68] used a value of 0.1 to clamp the particle's velocities, whereas Marini and Walczak [69] suggested anywhere between [0.0, 0.5]. However, V_{max} is problem-dependent and thus a sensitivity analysis will need to be conducted to determine the optimal value [70]. A linearly decreasing clamping-limit (LDCL) was suggested by a couple of studies [42,71]. Barrera et al. [71] tested various polynomial functions that reduce V_{max}, and found that a parabola function that orients particles towards a more localized search proved successfully:

$$\boldsymbol{V}_{max} = -\left(\frac{k}{Max_k} - 1\right)^n \times \left(\frac{\boldsymbol{x}_{max} - \boldsymbol{x}_{min}}{2}\right) \tag{18}$$

where, when, n = 3 or 5 good performance was reported by Barrera et al. Ratnaweera et al. [18] employed a velocity reinitialization scheme according to a random probability to avoid stagnation, while Ghalia [72] utilized an active penalty scheme that reset the particle's velocity component responsible for driving the particle outside the design space. The sensitivity analysis of $\boldsymbol{V}_{max}$ is studied, here, with different clamping constants [0.0, 1.0]. Furthermore, the local search enhancer (LSE) [71], the velocity clamping-limit decreasing method (VCLDM), the reinitialization method (RM) [18] and the active penalty scheme (APS) [72] are also considered. The results produced show that the clamping-limits have minimal impact on the algorithm's performance; LSE, LDCL, VCLDM and RM performed in a likewise manner. However, this behaviour might be due to the nature of the problem as the maximum velocity is commonly set to the dynamic range of the design variables [10,49]. The empirical results of the sensitivity analysis of the clamping-limit on the PSO algorithm are displayed in Table 5. The convergence history displayed in Fig. 9 demonstrate that the different velocity clamping strategies resulted in a very similar performance with respect to particle navigation.

Table 6
Description of the considered state-of-the-art metaheuristic methods.

Method	Description
Adaptive Firefly Algorithm (AFA) [73]	Firefly Algorithm (FA) simulates the behaviour of the flashing of fireflies. To improve, an adaptive FA (AFA) was proposed where both adaptive and self-adaptive parameters were utilized. The penalty function approach was employed for handling constraints.
Mine Blast Algorithm (MBA) [74]	MBA simulates the explosion of mine bombs and aims to locate the mine that would create the biggest explosion. Moreover, a set of feasibility rules are utilized for handling constraints.
Cuckoo Search (CS) [75]	CS simulates the obligate brood parasitic behaviour of some cuckoo species in combination with the Lévy flight behaviour of some birds and fruit flies.
Krill Herd (KH) [76]	KH simulates the behaviour of krill individuals, where the global minimum is located by determining the minimum distance to the food and the highest density of the krill swarm. The penalty function approach was employed for handling constraints.

Table 7
Best solution and statistical performance of the PSO-OPS and the considered algorithms for the speed-reducer design problem.

D.V.	AFA [73]	MBA [74]	CS [75]	KH [76]	Original PSO	Present work
x_1	3.500 E + 00	3.500 E + 00	3.501 E + 00	3.500 E + 00	3.508 E + 00	3.500 E + 00
x_2	0.700 E + 00	0.700 E + 00	0.700 E + 00	0.700 E + 00	0.700 E + 00	0.700 E + 00
x_3	1.700 E + 01	1.700 E + 01	1.700 E + 01	1.700 E + 01	1.700 E + 01	1.700 E + 01
x_4	7.302 E + 00	7.300 E + 00	7.605 E + 00	7.366 E + 00	7.300 E + 00	7.300 E + 00
x_5	7.800 E + 00	7.716 E + 00	7.818 E + 00	7.823 E + 00	8.300 E + 00	7.715 E + 00
x_6	3.350 E + 00	3.350 E + 00	3.352 E + 00	3.350 E + 00	3.365 E + 00	3.350 E + 00
x_7	5.287 E + 00	5.287 E + 00	5.287 E + 00	5.287 E + 00	5.290 E + 00	5.287 E + 00
Best	2.996 E + 03	2.994 E + 03	3.001 E + 03	2.997 E + 03	3.012 E + 03	2.994 E + 03
Worst	2.997 E + 03	3.000 E + 03	3.009 E + 03	3.011 E + 03	3.150 E + 03	2.994 E + 03
Mean	2.996 E + 03	2.997 E + 03	3.007 E + 03	3.006 E + 03	3.051 E + 03	2.994 E + 03
SD	9.000 E-02	1.560 E + 00	4.968 E + 00	2.638 E + 00	1.807 E + 01	1.583 E-07

4. Verification of sensitivity analysis

The results of the sensitivity analysis are verified by comparing the performance of PSO with other metaheuristic methods from the literature. Since a large swarm size and a maximum number of iterations increase computational efforts, a population of 50 particles and iteration of 500 are set to reduce function evaluations. The best performing acceleration coefficients combination ($c_1 = 2.5$, $c_2 = 1.0$) and inertia weight ($w = 0$) are employed in this verification study, this parameter set is hereby referred to as PSO-OPS (PSO-Optimal Parameter Set). Similar to the statistical approach taken in the sensitivity analysis, the best, worst, and mean solution, and standard deviation are considered for comparison. The speed-reducer, the tension/compression spring and the three-bar truss constrained optimization problems are used in this study, and run 30 times to extract statistical data. The problem descriptions are given in the Appendix. Five metaheuristic methods, Adaptive Firefly Algorithm (AFA) [73], Mine Blast Algorithm (MBA) [74], Cuckoo Search (CS) [75], Krill Herd (KH) [76], and PSO [1], are used for the verification study and their descriptions are given in Table 6.

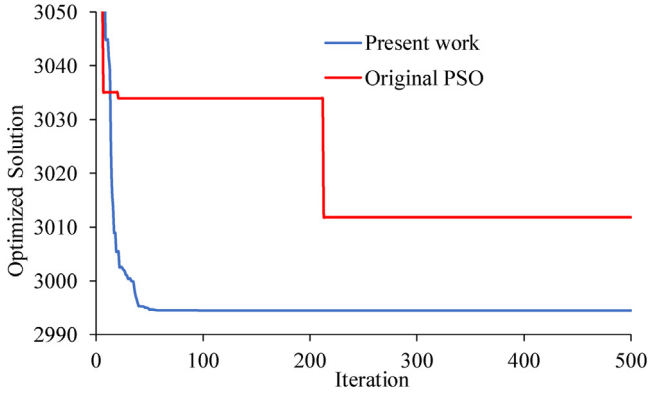

Fig. 10. Comparison of the convergence of PSO-OPS and the original PSO for the speed-reducer design problem.

4.1. Speed-Reducer

The speed-reducer problem considered in the sensitivity analysis is reintroduced in this algorithm comparative study. The best solution obtained with PSO-OPS is $x = [3.500000,\ 0.700000,\ 17.000000,\ 7.300000,\ 7.715320,\ 3.350215,\ 5.286654]$ with $f(x) = 2.994\,E + 04$. Comparison between the statistical performance and the best solutions obtained by the PSO-OPS and the other considered metaheuristic algorithms is presented in Table 7. The considered algorithms are AFA, MBA, CS, KH and PSO. In terms of obtaining the best solution, MBA, and PSO-OPS managed to obtain the best solution, $f(x) = 2994.471066$, while AFA, CS, KH, and PSO could not. As for the quality of results, PSO-OPS outperformed the considered methods by producing the lowest worst and mean solution, and standard deviation. Fig. 10 depicts the convergence curves of PSO-OPS and the original PSO for the speed-reducer problem, where the optimized solution refers to the best-obtained solution by the swarm.

4.2. Tension/Compression spring

The tension/compression spring design problem consists of a nonlinear objective function with four nonlinear inequality equations [77] and is illustrated in Fig. 11. The objective of this problem is to minimize the structural weight of the coil spring, and it consists of three continuous design variables. The design variables are the wire diameter $d\ (x_1)$, mean coil diameter $D\ (x_2)$, and the number of active coils $N\ (x_3)$. Furthermore, the spring is subjected to constraints on minimum deflection, shear stress, surge frequency, and the outer diameter of the spring. The best solution obtained with PSO-OPS is $x = [0.051704,\ 0.357071,\ 11.268276]$ with $f(x) = 1.267\,\mathrm{E} - 02$.

Comparison between the statistical performance and the best solutions obtained by the PSO-OPS and the other considered metaheuristic algorithms is presented in Table 8. The considered algorithms are AFA, MBA, KH and PSO. In terms of obtaining the best solution, AFA, MBA, and PSO-OPS managed to obtain the best solution, $f(x) = 0.012665$, while KH, and PSO could not. As for

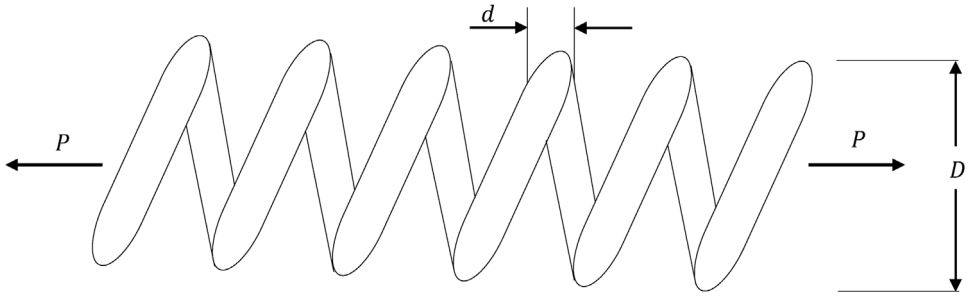

Fig. 11. Schematic representation of the tension/compression spring design problem.

Table 8
Best solution and statistical performance of the PSO-OPS and the considered algorithms for the tension/compression spring design problem.

D.V.	AFA [73]	MBA [74]	KH [76]	Original PSO	Present work
x_1	5.167 E-02	5.166 E + 00	5.166 E + 00	5.000 E-02	5.170 E-02
x_2	3.562 E-01	3.559 E + 00	3.559 E + 00	3.174 E-01	3.571 E-01
x_3	1.132 E + 01	1.134 E + 01	1.134 E + 01	1.404 E + 01	1.127 E + 01
Best	1.267 E-02	1.267 E-02	1.268 E-02	1.273 E-02	1.267 E-02
Worst	1.271 E-02	1.290 E-02	1.290 E-02	3.046 E-02	1.616 E-02
Mean	1.268 E-02	1.271 E-02	1.270 E-02	1.389 E-02	1.335 E-02
SD	1.281 E-05	6.300 E-05	3.000 E-04	3.398 E-03	7.994 E-04

Table 9
Best solution and statistical performance of the PSO-OPS and the considered algorithms for the three-bar truss design problem.

D.V.	KH [76]	MBA [74]	CS [75]	Original PSO	Present work
x_1	7.885 E-01	7.886 E-01	7.887 E-01	7.881 E-01	7.886 E-01
x_2	4.088 E-01	4.086 E-01	4.090 E-01	4.099 E-01	4.082 E-01
Best	2.639 E + 02	2.639 E + 02	2.640 E + 02	2.639 E + 02	2.639 E + 02
Worst	2.650 E + 02	2.639 E + 02	N/A	2.639 E + 02	2.639 E + 02
Mean	2.639 E + 02	2.639 E + 02	2.641 E + 02	2.639 E + 02	2.639 E + 02
SD	1.658 E-01	3.930 E-03	9.000 E-5	9.904 E-02	1.354 E-03

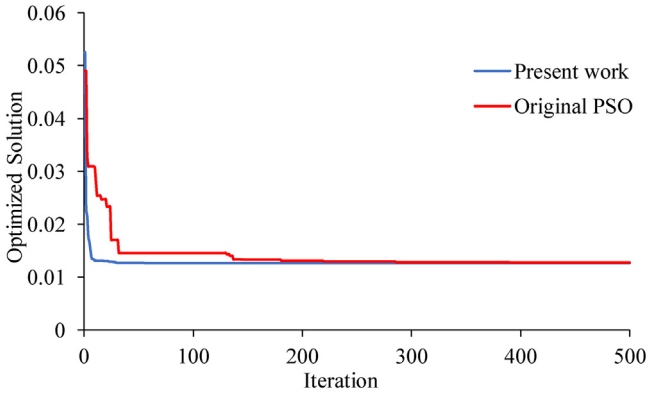

Fig. 12. Comparison of the convergence of PSO-OPS and the original PSO for the speed-reducer design problem.

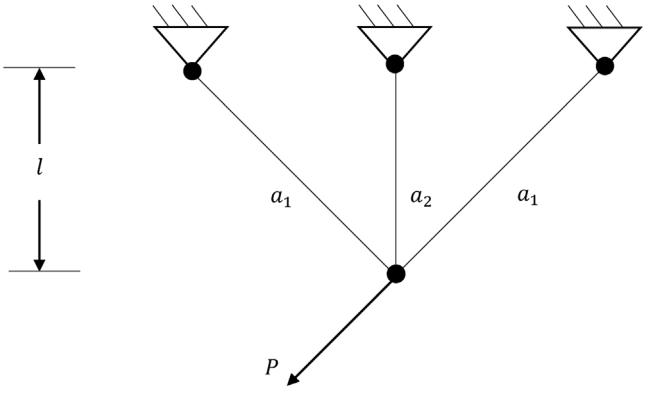

Fig. 13. Schematic representation of the three-bar truss design problem.

the quality of results, PSO-OPS was outperformed by the considered methods. However, despite PSO-OPS weaker performance, it managed to outperform PSO and produced a better statistical performance. Fig. 12 depicts the convergence curves of PSO-OPS and the original PSO for the speed-reducer problem, where the optimized solution refers to the best-obtained solution by the swarm.

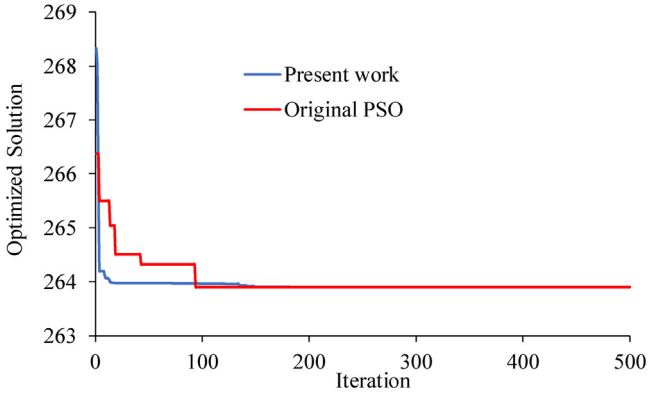

Fig. 14. Comparison of the convergence of PSO-OPS and the original PSO for the speed-reducer design problem.

4.3. Three-bar truss

The three-bar truss design problem consists of a nonlinear objective function with three nonlinear inequality equations [77] and is illustrated in Fig. 13, where l is the length of the bar (and P is the applied load. The objective of this problem is to minimize the structural weight of the truss, and it consists of two continuous design variables. The design variables are the cross-sectional area of the bars a_1 and a_2 (x_1 and x_2). The best solution obtained with PSO-OPS is $x = [0.788677,\ 0.408244]$ with $f(x) = 2.639\,E + 02$. Comparison between the statistical performance and the best solutions obtained by the PSO-OPS and the other considered metaheuristic algorithms is presented in Table 9. The considered algorithms are KH, MBA, CS and PSO. In terms of obtaining the best solution, KH, MBA, PSO, and PSO-OPS managed to obtain the best solution, $f(x) = 2.639\,E + 02$, while CS could not. As for the quality of results, PSO-OPS outperformed the considered methods by producing the lowest worst and mean solution, and standard deviation. Fig. 14 depicts the convergence curves of PSO-OPS and the original PSO for the speed-reducer problem, where the optimized solution refers to the best-obtained solution by the swarm.

5. Conclusion

Particle Swarm Optimization method may suffer from premature convergence and sensitivity to the initial control settings. An extensive sensitivity analysis was conducted to study the impact of the control parameters, and to determine the optimal parameter combination. A constrained optimization problem was considered for the parameter analysis, and the results revealed that PSO was most sensitive to the inertia weight and the acceleration coefficients. The optimal parameter combination was verified through a comparative study with other metaheuristic algorithms by solving three constrained design problems. The verification study revealed that PSO with the proposed parameters managed to outperform the other considered methods in all but one case, where it performed competitively.

The next step in this line of research will be to conduct a more comprehensive parameter analysis by studying different combinations simultaneously, as opposed to the proposed methodology of one-at-a-time sensitivity analysis. The applied method has its merit since it allows the user to determine the parameter responsible for failure. The applied constrained technique will need to be considered as well since it affects particle navigation, and could potentially further improve the performance of PSO.

Declaration of Competing Interest

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors

Acknowledgment

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors

Appendix A. Benchmark Constrained Optimization Problems

A.1. Speed Reducer

$$minf(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

$$c(x)_1 = \frac{27}{x_1x_2^2x_3} - 1 \le 0$$

$$c(x)_2 = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0$$

$$c(x)_3 = \frac{1.93x_4^3}{x_2x_6^4x_3} - 1 \le 0$$

$$c(x)_4 = \frac{1.93x_5^3}{x_2x_6^4x_3} - 1 \le 0$$

$$c(x)_5 = \frac{\sqrt{745\left(\frac{x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6}}{110x_6^3} - 1 \le 0$$

$$c(x)_6 = \frac{\sqrt{745\left(\frac{x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6}}{85x_7^3} - 1 \le 0$$

$$c(x)_7 = \frac{x_2x_3}{40} - 1 \le 0$$

$$c(x)_8 = \frac{5x_2}{x_1} - 1 \le 0$$

$$c(x)_9 = \frac{x_1}{12x_2} - 1 \le 0$$

$$c(x)_{10} = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0$$

$$c(x)_{11} = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0$$

where $2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8$, $17 \le x_3 \le 28$, $7.3 \le x_4 \le 8.3$, $7.3 \le x_5 \le 8.3$, $2.9 \le x_6 \le 3.9$ and $5.0 \le x_7 \le 5.5$.

A.2. Tension/Compression Spring

$$minf(x) = (x_3 + 2)x_2x_1^2$$

$$c(x)_1 = 1 - \frac{x_2^3x_3}{71785x_1^4} \le 0$$

$$c(x)_2 = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \le 0$$

$$c(x)_3 = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0$$

$$c(x)_4 = \frac{x_2 + x_1}{1.5} - 1 \le 0$$

where $0.05 \le x_1 \le 2.00$, $0.25 \le x_2 \le 1.30$ and $2.00 \le x_3 \le 15.00$.

A.3. Three-Bar Truss

$$minf(x) = 200\sqrt{2}x_1 + 100x_2$$

$$c(x)_1 = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} - 1 \le 0$$

$$c(x)_2 = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} - 1 \le 0$$

$$c(x)_3 = \frac{1}{\sqrt{2}x_2 + x_1} - 1 \le 0$$

where $0.0 \le x_1, x_2 \le 1.0$.

References

[1] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, Perth, Australia, 1995.
[2] S. Ganguly, N. Sahoo, D. Das, Multi-objective particle swarm optimization based on fuzzy-Pareto-dominance for possibilistic planning of electrical distribution systems incorporating distributed generation, Fuzzy Sets Syst. 213 (2013) 47–73.
[3] Y. Cai, S.X. Yang, An improved PSO-based approach with dynamic parameter tuning for cooperative target searching of multi-robots, in: Proceedings of World Automation Congress, Waikoloa, HI, USA, 2013, pp. 616–621.
[4] P.N. Kechagiopoulos, G.N. Beligiannis, Solving the Urban Transit Routing Problem using a particle swarm optimization based algorithm, Appl. Soft Comput. 21 (2014) 654–676.
[5] M. Hasanipanah, D.J. Armaghani, H.B. Amnieh, M.Z. Abd Majid, M.M.D. Tahir, Application of PSO to develop a powerful equation for prediction of flyrock due to blasting, Neural Comput. Appl. 28 (1) (2017) 1043–1050.
[6] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.
[7] M.A. Panduro, C.A. Brizuela, L.I. Balderas, D.A. Acosta, A Comparison of Genetic Algorithms, Particle Swarm Optimization and the Differential Evolution

Method for the Design of Scannable Circular Antenna Arrays, Prog. Electromagn. Res. B 13 (2009) 171–186.
[8] Y. Wang, J. Lv, L. Zhu, Y. Ma, Crystal structure prediction via particle swarm optimization, Phys. Rev. B 82 (9) (2010) 94116–1–094116–94118.
[9] S. Pandey, L. Wu, S.M. Guru, R. Buyya, A particle swarm optimization-based heuristic for scheduling workflow applications in Cloud computing environments, in: IEEE International Conference on Advanced Information Networking and Applications, Perth, WA, 2010, pp. 400–407.
[10] X. Hu, R.C. Eberhart, Y. Shi, Engineering optimization with particle swarm, in: Proceedings of the IEEE Swarm Intelligence Symposium, Indianapolis, IN, 2003, pp. 53–57.
[11] F. van den Bergh, A. Engelbrecht, A New locally convergent particle swarm optimiser, in: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, Yasmine Hammamet, Tunisia, 2002, pp. 96–101.
[12] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: In IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence, Anchorage, Alaska, USA, 1998, pp. 69–73.
[13] M. Clerc, J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (1) (2002) 58–73.
[14] H. Wang, H. Sun, C. Li, S. Rahnamayan, J. Pan, Diversity enhanced particle swarm optimization with neighborhood search, Inf. Sci. 223 (2013) 119–135.
[15] A.H. Gandomia, G.J. Yun, X.-S. Yang, S. Talatahari, Chaos-enhanced accelerated particle swarm optimization, Commun. Nonlinear Sci. Numer. Simul. 18 (2) (2013) 327–340.
[16] Y. Feng, G.-F. Teng, A.-X. Wang, Y.-M. Yao, Chaotic inertia weight in particle swarm optimization, in: Second International Conference on Innovative Computing, Informatio and Control, Kumamoto, Japan, 2007, pp. 475–479.
[17] M. Pluhacek, R. Senkerik, D. Davendra, Z.K. Oplatkova, I. Zelinka, On the behavior and performance of chaos driven PSO algorithm with inertia weight, Comput. Math. With Appl. 66 (2) (2013) 122–134.
[18] A. Ratnaweera, S.K. Halgamuge, H.C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, IEEE Trans. Evol. Comput. 8 (3) (2004) 240–255.
[19] Z.-H. Zhan, J. Zhang, Y. Li, H.S.-H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. Part B 39 (6) (2009) 1362–1381.
[20] W.-N. Chen, J. Zhang, Y. Lin, N. Chen, Z.-H. Zhan, H.S.-H. Chung, Y. Li, Y.-H. Shi, Particle swarm optimization with an aging leader and challengers, IEEE Trans. Evol. Comput. 17 (2) (2013) 241–258.
[21] P.S. Shelokar, P. Siarry, V.K. Jayaraman, B.D. Kulkarni, Particle swarm and ant colony algorithms hybridized for improved continuous optimization, Appl. Math. Comput. 188 (1) (2007) 129–142.
[22] C.-F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 3 (2) (2004) 997–1006.
[23] D.J. Armaghani, R.S.N.S.B.R. Shoib, K. Faizi, A.S.A. Rashid, Developing a hybrid PSO–ANN model for estimating the ultimate bearing capacity of rock-socketed piles, Neural Comput. Appl. 28 (2) (2017) 391–405.
[24] F. Van den Bergh, A. Engelbrecht, A study of particle swarm optimization particle trajectories, Inf. Sci. 176 (8) (2006) 937–971.
[25] G. Ciuprina, D. Ioan, I. Munteanu, Use of intelligent-particle swarm optimization in electromagnetics, IEEE Trans. Magn. 38 (2) (2002) 1037–1040.
[26] T. Beielstein, K.E. Parsopoulos, M.N. Vrahatis, Tuning PSO parameters through sensitivity analysis, in: Interner Bericht Des Sonderforschungsbereichs (SFB) 531 Computational Intelligence No.CI-124/02, Affiliation: Universität Dortmund, 2002, January.
[27] R.V. O'Neill, R.H. Gardner, J.B. Mankin, Analysis of parameter error in a nonlinear model, Ecol. Modell. 8 (1980) 297–311.
[28] D.M. Hamby, A review of techniques for parameter sensitivity analysis of environmental models, Environ. Monit. Assess. 32 (2) (1994) 135–154.
[29] E. Mezura-Montes, C.A. Coello Coello, Constraint-handling in nature-inspired numerical optimization: past, present and future, Swarm Evol. Comput. 1 (4) (2011) 173–194.
[30] C.A. Coello Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Comput. Methods Appl. Mech. Eng. 191 (11–12) (2002) 1245–1287.
[31] K. Parsopoulos, M. Vrahatis, Unified particle swarm optimization for solving constrained engineering optimization problems, Lecture Notes in Computer Science (LNCS) 3612 (2005) 582–591.
[32] K.E. Parsopoulos, M.N. Vrahatis, Particle Swarm Optimization Metho d for Constrained Optimization Problems, Intelligent Technologies - Theory and Appli- cations: New Trends in Intelligent Technologies 76 (2002) 214–220.
[33] X. Hu, R. Eberhart, Solving constrained nonlinear optimization problems, in: Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics, Orlando, Florida, USA, 2002, pp. 203–206.
[34] Q. He, L. Wang, A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization, Appl. Math. Comput. 186 (2) (2007) 1407–1422.
[35] R. Eberhart, J. Kennedy, A New optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995, pp. 39–43.
[36] M.R. Bonyadi, Z. Michalewicz, Particle swarm optimization for single objective continuous space problems: a review, Evol. Comput. 25 (1) (2017) 1–54.
[37] A.P. Engelbrecht, Particle swarm optimization, in: Computational Intelligence: An Introduction, John Wiley & Sons, Hoboken, NJ, 2007, pp. 289–358.
[38] R. Poli, J. Kennedy, T. Blackwell, Particle Swarm Optimisation: An Overview, Swarm Intell. 1 (1) (2007) 33–57.
[39] P.N. Suganthan, Particle swarm optimiser with neighbourhood operator, in: Proceedings of the Congress on Evolutionary Computation, Washington, DC, USA, 1991, pp. 1958–1962.
[40] Y. Shi, R.C. Eberhart, Parameter selection in particle swarm optimization, in: Evolutionary Programming VII. Lecture Notes in Computer Science, Springer, Berlin, 1998, pp. 591–600.
[41] A. Banks, J. Vincent, C. Anyakoha, A Review of Particle Swarm Optimization. Part I: Background and Development, Nat. Comput. 6 (4) (2007) 467–484.
[42] A.P. Engelbrecht, Computational Intelligence: An Introduction, 2nd edition, Wiley, 2007.
[43] R.C. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, in: Proceedings of the Congress on Evolutionary Computation, Seoul, South Korea, 2001, pp. 81–86.
[44] J.S. Arora, Introduction to Optimum Design, 4th ed., Elsevier, Ed., Academic Press, 2017.
[45] L. dos Santos Coelho, Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems, Expert Syst. Appl. 37 (2010) 1676–1683.
[46] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Inf. Process. Lett. 85 (6) (2003) 317–325.
[47] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, Ieee Trans. Evol. Comput. 8 (3) (2004) 225–239.
[48] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of the Congress on Evolutionary Computation, La Jolla, CA, USA, 2000, pp. 84–88.
[49] R.C. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Proceedings of the Congress on Evolutionary Computation, Seoul, South Korea, 2001, pp. 94–97.
[50] Y. Shi, R.C. Eberhart, Fuzzy adaptive particle swarm optimization, in: Proceedings of the Congress on Evolutionary Computation, Seoul, South Korea, 2001, pp. 101–106.
[51] A. Chatterjee, P. Siarry, Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization, Comput. Oper. Res. 33 (3) (2006) 859–871.
[52] M. Taherkhani, R. Safabakhsh, A novel stability-based adaptive inertia weight for particle swarm optimization, Appl. Soft Comput. 38 (2016) 281–295.
[53] E. Mezura-Montes, C.A. Coello Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, Int. J. Gen. Syst. 37 (4) (2008) 443–473.
[54] Z. Michalewicz, M. Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evol. Comput. 4 (1) (1996) 1–32.
[55] J.T. Richardson, M.R. Palmer, G.E. Liepins, M. Hilliard, Some guidelines for genetic algorithms with penalty functions, in: Proceedings of the International Conference on Genetic Algorithms, San Mateo, CA, USA, 1989, pp. 41–49.
[56] J. Golinski, An adaptive optimization system applied to machine synthesis, Mech. Mach. Theory 8 (4) (1973) 419–436.
[57] E.C. Laskari, K.E. Parsopoulos, M.N. Vrahatis, Particle swarm optimization for integer programming, in: Proceedings of the Congress on Evolutionary Computation, Honolulu, HI, USA, 2002, pp. 1576–1581.
[58] H. Wang, Z. Hu, Y. Sun, Q. Su, X. Xia, Modified backtracking search optimization algorithm inspired by simulated annealing for constrained engineering optimization problems, Comput. Intell. Neurosci. 2018 (2018) 1–27.
[59] N.B. Guedria, Improved accelerated PSO algorithm for mechanical engineering optimization problems, Appl. Soft Comput. 40 (2016) 455–467.
[60] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the Congress on Evolutionary Computation, Washington, DC, USA, 1999, pp. 1945–1950.
[61] S.V.A.N.K. Abad, M. Yilmaz, D.J. Armaghani, A. Tugrul, Prediction of the durability of limestone aggregates using computational techniques, Neural Comput. Appl. 29 (2018) 423–433.
[62] J.C. Bansal, P.K. Singh, M. Saraswat, A. Verma, S.S. Jadon, A. Abraham, Inertia weight strategies in particle swarm optimization, in: World Congress on Nature and Biologically Inspired Computing, Salamanca, Spain, 2011, pp. 640–647.
[63] J. Kennedy, The behavior of particles, in: International Conference on Evolutionary Programming, San Diego, CA, USA, 1998, pp. 581–589.
[64] E. Ozcan, C. Mohan, Particle swarm optimization: surfing the waves, in: Proceedings of the IEEE Congress on Evolutionary Computation, Washington, DC, USA, 1999, pp. 1939–1944.
[65] J. Kennedy, The particle swarm: social adaptation of knowledge, in: Proceedings of IEEE International Conference on Evolutionary Computation, Indianapolis, IN, USA, 1997, pp. 303–308.
[66] M. Clerc, The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, IEEE Congress on Evolutionary Computation (1999) 1951–1957.
[67] Y.F. Yu, G. Li, C. Xu, An improved particle swarm optimization algorithm, Applied Mechancis and Materials 401–403 (2013) 1328–1335.
[68] F. Shahzad, A.R. Baig, S. Masood, M. Kamran, N. Naveed, Opposition-based particle swarm optimization with velocity clamping (OVCPSO), Advances in Computational Intelligence 116 (2009) 339–348.
[69] F. Marini, B. Walczak, Particle swarm optimization (PSO). A tutorial, Chemom. Intell. Lab. Syst. 149 (2015) 153–165.

[70] A. Engelbrecht, Particle swarm optimization: velocity initialization, in: Proceedings of the IEEE Congress on Evolutionary Computation, Brisbane, QLD, Australia, 2012, pp. 1–8.
[71] J. Barrera, O. Álvarez-Bajo, J.J. Flores, C.A. Coello Coello, Limiting the velocity in the particle swarm optimization algorithm, Comput. Y Sist. 20 (4) (2016) 635–645.
[72] M. Ben Ghalia, Particle swarm optimization with an improved exploration-exploitation balance, in: Midwest Symposium on Circuits and Systems, Knoxville, TN, USA, 2008, pp. 759–762.
[73] A. Baykasoglu, F.B. Ozsoydan, Adaptive firefly algorithm with chaos for mechanical design optimization problems, Appl. Soft Comput. 36 (2015) 152–164.
[74] A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems, Appl. Soft Comput. 13 (5) (2013) 2592–2612.
[75] A.H. Gandomi, X.-S. Yang, A.H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems, Eng. Comput. 29 (2013) 17–35.
[76] A.H. Gandomi, A.H. Alavi, An introduction of krill herd algorithm for engineering optimization, J. Civ. Eng. Manag. 22 (3) (2016) 302–310.
[77] J.S. Arora, Introduction to Optimum Design, 1st ed., McGraw-Hill, New York, 1989.

Mewael Isiet is a Ph.D. student of Mining Engineering at the University of British Columbia in Vancouver, Canada. Previously, Mewael Isiet completed his MASc of Mechanical Engineering, under the supervision of Dr. Mohamed Gadala, at the University of British Columbia in 2019. His current research interests include design optimization, model-based control, sensitivity analysis and mineral processing.

Dr. Gadala is a Professor Emeritus of Mechanical Engineering at the University of British Columbia in Vancouver, Canada. Also, Dr. Gadala is the Chair of Mechanical Engineering Department at Abu Dhabi University in UAE. His current research interests include finite element and numerical simulation of structural & CFD problems; online monitoring of rotating equipment & turbo machinery; inverse heat transfer analysis for cooling on runout table, fracture mechanics and design optimization. Dr. Gadala is the recipient of the NSERC University-Industry Synergy award; Patrick Campbell chair in design in Mechanical Engineering at UBC for 12 years; Established PACE (Partner for Advancement of CAE Education) lab in UBC with multi million dollars industrial contributions of hardware & software licenses. Prof. Gadala has authored & co-authored over 180 refereed papers, 4 book chapters, 4 patents and 4 software manuals. Before joining academia, Prof. Gadala has worked in various industries in the US and Canada for more than twelve years and he also taught at the Univ. of Michigan-Dearborn.