Chapter 2
Particle Swarm Optimization

2.1 Introduction

Particle swarm optimization (PSO) algorithms are nature-inspired population-based metaheuristic algorithms originally accredited to Eberhart, Kennedy, and Shi [1, 2]. These algorithms mimic the social behavior of birds flocking and fishes schooling. Starting form a randomly distributed set of particles (potential solutions), the algorithms try to improve the solutions according to a quality measure (fitness function). The improvisation is performed through moving the particles around the search space by means of a set of simple mathematical expressions which model some interparticle communications. These mathematical expressions, in their simplest and most basic form, suggest the movement of each particle toward its own best experienced position and the swarm's best position so far, along with some random perturbations. There is an abundance of different variants using different updating rules, however.

Though being generally known and utilized as an optimization technique, PSO has its roots in image rendering and computer animation technology where Reeves [3] defined and implemented a particle system as a set of autonomous individuals working together to form the appearance of a fuzzy object like a cloud or an explosion. The idea was to initially generate a set of points and to assign an initial velocity vector to each of them. Using these velocity vectors, each particle changes its position iteratively while the velocity vectors are being adjusted by some random factors.

Reynolds [4] added the notion of inter-object communication to Reeves' particle system to introduce a flocking algorithm in which the individuals were able to follow some basic flocking rules such as trying to match each other's velocities. Such a system allowed for modeling more complex group behaviors in an easier and more natural way.

Kennedy and Eberhart [1] while trying to "graphically simulate the graceful but unpredictable choreography of a bird flock" came across the potential optimization

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*,
DOI 10.1007/978-3-319-46173-1_2

capabilities of a flock of birds. In the course of refinement and simplification of their paradigm, they discussed that the behavior of the population of agents that they were suggesting follows the five principles of *swarm* intelligence articulated by Millonas [5]. First is the proximity principle: the population should be able to carry out simple space and time computations. Second is the quality principle: the population should be able to respond to quality factors in the environment. Third is the principle of diverse response: the population should not commit its activities along excessively narrow channels. Fourth is the principle of stability: the population should not change its mode of behavior every time the environment changes. Fifth is the principle of adaptability: the population must be able to change behavior mode when it is worth the computational price. They also mention that they compromisingly call their massless volume-less population members *particles* in order to make the use of concepts like velocity and acceleration more sensible. Thus, the term particle swarm optimization was coined.

2.2 PSO Algorithm

2.2.1 Development

As Kennedy and Eberhart [1] indicated appropriately particle swarm optimization is probably best presented and understood by explaining its conceptual development. Hence, the algorithm's transformation process from its earliest stages to its current canonical form is briefly reviewed in this section. Future discussion on the main aspects and issues would be more easily done this way.

The earliest attempt to use the concept for social behavior simulation carried out by Kennedy and Eberhart [1] resulted in a set of agents randomly spread over a torus pixel grid which used two main strategies: nearest neighbor velocity matching and craziness. At each iteration, a loop in the program is determined for each agent which other agent was its nearest neighbor, then assigned that agent's X and Y velocities to the agent in focus. As it is predictable, it has been viewed that sole use of such a strategy will quickly settle down the swarm on a unanimous, unchanging direction. To avoid this, a stochastic variable called craziness was introduced. At each iteration, some change was added to randomly chosen X and Y velocities. This introduced enough variation into the system to give the simulation a "life-like" appearance. The above observation points out one of the most necessary features of PSO which indicates its seemingly unalterable nondeterministic nature: incorporation of randomness.

Kennedy and Eberhart took the next step by replacing the notion of "roost" (a place that the birds know previously) in Heppner and Grenander [6] by "food" (for which the birds must search) and therefore converted the social simulation algorithm into an optimization paradigm. The idea was to let the agents (birds) find an unknown favorable place in the search space (food source) through capitalizing

on one another's knowledge. Each agent was able of remembering its best position and knowing the best position of the whole swarm. The extremum of the mathematical function to be optimized can be thought of as the food source. After a series of minor alterations and elimination of the ancillary variables, the updating rules for calculating the next position of a particle were introduced as:

$$v_{i,j}^{k+1} = v_{i,j}^{k} + c_1 r_1 \left(xbest_{i,j}^{k} - x_{i,j}^{k} \right) + c_2 r_2 \left(xgbest_{j}^{k} - x_{i,j}^{k} \right) \tag{2.1}$$

$$x_{i,j}^{k+1} = x_{i,j}^{k} + v_{i,j}^{k+1} \tag{2.2}$$

where $x_{i,j}^{k}$ and $v_{i,j}^{k}$ are the jth component of the *i*th particle's position and velocity vector, respectively, in the *k*th iteration; r_1 and r_2 are two random numbers uniformly distributed in the range (1,0); $xbest_i$ and xgbest indicate the best positions experienced so far by the *i*th particle and the whole swarm, respectively; and c_1 and c_2 are two parameters representing the particle's confidence in itself (cognition) and in the swarm (social behavior), respectively. These two parameters were set equal to 2 in the initial version presented by Kennedy and Eberhart [1] so that the particles would overfly the target about half the time. These two parameters are among the most important parameters of the algorithm in that they control the balance between exploration and exploration tendencies. A relatively high value of c_1 will encourage the particles to move toward their local best experiences, while higher values of c_2 will result in faster convergence to the global best position.

Although the above formulation embodies the main concept of PSO that has survived over time and forms the skeleton of quite all subsequent variants, it has still been subject to amendment. Eberhart et al. [7] introduced a maximum velocity parameter, V_{max}, in order to prevent particles from leaving the search space. Shi and Eberhart [8] discussed the role of the three terms of Eq. (2.1) and concluded that the first term, previous velocity of the particle, has an important effect on global and local search balance. By eliminating this term, the particles cannot leave their initially encircled portion of the search space, and the search space will shrink gradually over time. This will be equivalent to a local search procedure. Alternatively, by giving the previous velocity term relatively higher effects, the particles will be reluctant to converge to the known good positions. They will instead tend to explore unseen regions of the search space. This could be conceived as global search tendency. Both the local search and global search will benefit solving some kinds of problems. Therefore, an inertia weight w is introduced into Eq. (2.1) in order to maintain balance between these two effects:

$$v_{i,j}^{k+1} = w v_{i,j}^{k} + c_1 r_1 \left(xlbest_{i,j}^{k} - x_{i,j}^{k} \right) + c_2 r_2 \left(xgbest_{j}^{k} - x_{i,j}^{k} \right) \tag{2.3}$$

Shi and Eberhart [8] indicated that the inertia weight can be a positive constant or even a positive linear or nonlinear function of time. They examined the use of constant values in the range [0, 1.4] for the benchmark problem of Schaffer's f6 function and concluded the range [0.9, 1.2] to be appropriate. Later, Eberhart and

Shi [9] indicated that the use of the inertia weight w, which decreases linearly from about 0.9 to 0.4 during a run, provides improved performance in a number of applications. Many different research works have focused on inertia weight parameter, and different strategies have been proposed ever since. A brief discussion on these methods and strategies will be presented in the next section.

Later, Clerc [10] indicated that the use of a constriction factor may be necessary to insure convergence of the particle swarm algorithm and proposed an alternative formulation for the velocity vector:

$$v_{i,j}^{k+1} = \chi\left[v_{i,j}^{k} + c_1 r_1\left(xlbest_{i,j}^{k} - x_{i,j}^{k}\right) + c_2 r_2\left(xgbest_{j}^{k} - x_{i,j}^{k}\right)\right] \tag{2.4}$$

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad where \quad \varphi = c_1 + c_2, \quad \varphi > 4 \tag{2.5}$$

Schematic movement of a particle is illustrated in Fig. 2.1.

Such a formulation was intended to impose some restriction on velocity vectors and thus to prevent divergence. Eberhart and Shi [9] compared the use of inertia weights and constriction factors in particle swarm optimization and concluded that the two approaches are equivalent and could be interchangeably used by proper parameter setting. They also indicated that the use of constriction factor does not eliminate the need for V_{max} parameter unlike what might be assumed at the first glance. Though the two approaches are shown to be equivalent, they both survived and have been continually used by researchers. Simultaneous utilization of inertia weight and constriction factor can also be found in the literature (e.g., see [11] among others).

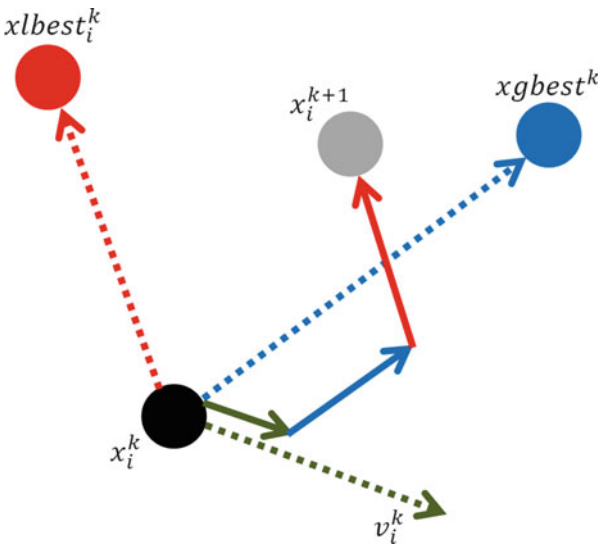

Fig. 2.1 Schematic movement of a particle based on Eq. (2.4)

2.2.2 PSO Algorithm

The general structure of a canonical PSO algorithm is as follows [12]:

```
procedure Particle Swarm Optimization
begin
  Initialize x_i, v_i and xbest_i for each particle i;
  while (not termination condition) do
  begin
    for each particle i
      Evaluate objective function;
      Update xbest_i
    end
    for each i
      Set g equal to index of neighbor with best xbest_i;
      Use g to calculate v_i;
      Update x_i = x_i + v_i;
      Evaluate objective function;
       Update xbest_i
    end
  end
end
```

2.2.3 Parameters

Like any other metaheuristic algorithm, PSO's performance is dependent on the values of its parameters. The optimal values for the parameters depend mainly on the problem at hand and even the instance to deal with and on the search time that the user wants to spend in solving the problem [13]. In fact the main issue is to provide balance between exploration and exploitation tendencies of the algorithm.

Total number of particles, total number of iterations, inertia weight and/or constriction factor, and cognition and social behavior coefficients (c_1 and c_2) are the main parameters that should be considered in a canonical PSO. The total number of iterations could be replaced with a desired precision or any other termination criterion. In general, the search space of an n-dimensional optimization problem can be conceived as an n-dimensional hypersurface. The suitable values for a metaheuristic's parameters depend on relative ruggedness and smoothness of this hyperspace. For example, it is imaginable that in a smoother hyperspace, fewer number of particles and iteration numbers will be required. Moreover, in a smoother search space, there will be fewer local optimal positions and less exploration effort

will be needed, while in a rugged search space, a more through exploration of the search space will be advisable.

Generally speaking, there are two different strategies for parameter value selection, namely, off-line parameter initialization and online parameter tuning [13]. In off-line parameter initialization, the values of different parameters are fixed before the execution of the metaheuristic. These values are usually decided upon through empirical study. It should be noted that deciding about a parameter of a metaheuristic algorithm while keeping the others fixed (i.e., one-by-one parameter selection) may result in misleading observations since the interactions of the parameters are not taken into account. However, it is the common practice in the literature since examining combinations of the algorithm parameters might be very time-consuming. To perform such an examination, when desired, a meta-optimization approach may be performed, i.e., the algorithm parameters can be considered as design variables and be optimized in an overlying level.

The main drawback of the off-line approaches is their high computational cost since the process should be repeated for different problems and even for different instances of the same problem. Moreover, appropriate values for a parameter might change during the optimization process. Hence, online approaches that change the parameter values during the search procedure must be designed. Online approaches may be classified in two main groups [13]: dynamic approaches and adaptive approaches. In a dynamic parameter updating approach, the change of the parameter value is performed without taking into account the search progress. The adaptive approach changes the values according to the search progress.

Attempts have been made to introduce guidelines and strategies for selection of PSO parameters. Shi and Eberhart [14] analyzed the impact of inertia weight and maximum allowable velocity on the performance of PSO and provided some guidelines for selecting these two parameters. For this purpose, they utilized different combinations of w and Vmax parameters to solve the Schaffer's f6 test function while keeping other parameters unchanged. They concluded that when Vmax is small (<=2 for the f6 function), an inertia weight of approximately 1 is a good choice, while when Vmax is not small (>=3), an inertia weight $w = 0.8$ is a good choice. They declared that in absence of proper knowledge regarding the selection of Vmax, it is also a good choice to set Vmax equal to Xmax, and an inertia weight $w = 0.8$ is a good starting point. Furthermore, if a time-varying inertia weight is employed, even better performance can be expected. As the authors indicated, such an empirical approach using a small benchmark problem cannot be easily generalized.

Carlisle and Dozier [15] proposed another set of guidelines based on evidence from six experiments. They recommended to start with an asynchronous constricted algorithm setting $r_1 = 2.8$ and $r_2 = 1.3$. However, no directives are provided in order to progress from this initial setting.

Trelea [16] used dynamic system theory for a theoretical analysis of the algorithm producing some graphical guidelines for parameter selection. A simplified deterministic one-dimensional PSO was used for this study. Trelea indicates that

the results are predictably dependent on the form of the objective function. The discussion is supported by experiments on five benchmark functions.

Zhang et al. [17] suggested some optimal ranges for constriction factor and Vmax to Xmax ratio parameters based on experimental study on nine mathematical functions. The optimal range for constriction factor is claimed to be [4.05, 4.3], while for Vmax to Xmax ratio, the range [0.01, 1] is recommended.

More recently, Pedersen [18] carried out meta-optimization to tune the PSO parameters. A table is presented to help the practitioner choose appropriate PSO parameters based on the dimension of the problem at hand and the total number of function evaluations that is intended to be performed. Performance evaluation of PSO is performed using some mathematical functions. As mentioned before, the results of the abovementioned off-line parameter tuning studies are all problem dependent and could not be claimed as universally optimal.

Many different online tuning strategies are also proposed for different PSO parameters. For inertia weight, methods such as random inertia weight, adaptive inertia weight, sigmoid increasing/decreasing inertia weight, linear decreasing inertia weight, chaotic inertia weight and chaotic random inertia weight, oscillating inertia weight, global–local best inertia weight, simulated annealing inertia weight, natural exponent inertia weight strategy, logarithm decreasing inertia weight, and exponent decreasing inertia weight are reported in the literature. All of these methods replace the inertia weight parameter with a mathematical expression which is either dependent on the state of the search process (e.g., global best solution, current position of the particle, etc.) or not. Bansal et al. [19] examined the abovementioned inertia weight strategies for a set of five mathematical problems and concluded that chaotic inertia weight is the best strategy for better accuracy, while random inertia weight strategy is best for better efficiency. This shows that the choice of a suitable inertia weight strategy depends not only on the problem under consideration but also on the practitioner's priorities.

Other adaptive particle swarm optimization algorithms could be found in the literature [20].

2.2.4 Premature Convergence

One of the main advantages of PSO is its ability to attain reasonably good solutions relatively fast. At the same time, this is probably the algorithm's most recognized drawback. In fact, Angeline [21] while studying PSO versus evolutionary optimization techniques showed that although PSO discovers good quality solutions much faster than evolutionary algorithms, it does not improve the quality of the solutions as the number of generations is increased. This is because of the particles getting clustered in a small region of the search space and thus the loss of diversity [22]. Improving the exploration ability of PSO has been an active research topic in recent years [20].

Attempts have been made in order to improve the algorithm's exploration capabilities and thus to avoid premature convergence. van den Bergh and Engelbrecht [23] introduced a guaranteed convergence PSO (GCPSO) in which particles perform a random search around xgbest within a radius defined by a scaling factor. The algorithm is reported to perform better than original PSO in unimodal problems while producing similar results in multimodal ones. The scaling factor however is another parameter for which prior knowledge may be required to be optimally set.

Krink et al. [24] proposed a collision-free PSO where particles attempting to gather about a suboptimal solution bounce away. A random direction changer, a realistic bounce, and a random velocity changer were used as three bouncing strategies. The latter two are reported to significantly improve the exploration capabilities of the algorithm and obtain better results especially in multimodal problems.

Implementing diversity measures is another way to control swarm stagnation. Riget and Vesterstrøm [25] utilized such a measure along with alternative attraction and repulsion of the particles to and from the swarm best position. Repulsion could be induced by inverting the velocity update rule. The approach improves the performance in comparison to canonical PSO, especially when problems under consideration are multimodal.

Silva et al. [26] introduced a predator–prey optimization system in which a predator particle enforces other particles to leave the best position of the search space and explore other regions. Improved performance is reported based on experiments carried out on four high-dimensional test functions.

Jie et al. [27] introduced an adaptive PSO with feedback control of diversity in order to tune the inertia weight parameter and alleviate the premature convergence. The improvements increase the robustness and improve the performance of the standard PSO in multimodal functions.

Wang et al. [20] proposed a self-adaptive learning-based particle swarm optimization which used four PSO-based search strategies on probabilistic basis according to the algorithm's performance in previous iterations. The use of different search strategies in a learning-based manner helps the algorithm to handle problems with different characteristics at different stages of optimization process. Twenty-six test functions with different characteristics such as unimodality, multimodality, rotation, ill-condition, mis-scale, and noise are considered, and the results are compared with eight other PSO variants.

Kaveh and Zolghadr [28] introduced a democratic particle swarm optimization (DPSO) which derives the updating information of a particle from a more diverse set of sources instead of using local and global best solutions merely. An eligibility parameter is introduced which determines which particles to incorporate when updating a specific particle. The improved algorithm is compared to the standard one for some mathematical and structural problems. The performance is improved in the problems under consideration.

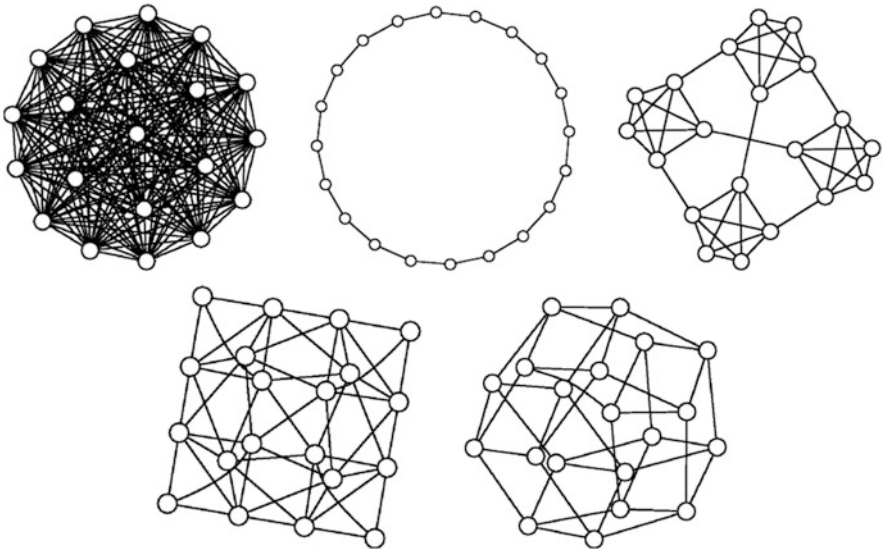

Fig. 2.2 Some topologies for PSO neighborhoods [29]. Fully connected, where all vertexes are connected to every other; *ring*, where every vertex is connected to two others; *four clusters*, with four cliques connected among themselves by gateways; *pyramid*, a triangular wire-frame pyramid; and *square*, which is a mesh where every vertex has four neighbors that wrap around on the edges as a torus

2.2.5 Topology

While xgbest of Eq. (2.1) is considered to be the whole swarm's global best position in canonical PSO, this is not necessarily always the case. Different topologies have been defined and used for interparticle communications in PSO [29, 30]. In fact in updating a particle's position, xgbest could mean the best particle position of a limited neighborhood to which the particle is connected instead of the whole swarm. It has been shown that the swarm topologies in PSO can remarkably influence the performance of the algorithm. Figure 2.2 shows some of the basic PSO neighborhood topologies introduced by Mendes et al. [29]. Many other topologies can be defined and used.

These different topologies affect the way that information circulates between the swarm's particles and thus can control exploration–exploitation behavior and convergence rate of the algorithm. Canonical PSO uses the fully connected topology in which all of the particles are neighbors. Such a topology exhibits a fast (and probably immature) convergence since all of the particles are directly linked to the global best particle and simultaneously affected by it. Thus, the swarm does not explore other areas of the search space and would most probably get trapped in local optima.

Ring topology which is a usual alternative to fully connected topology represents a regular graph with the minimum node degrees. This could be considered the

slowest way of information circulation between the particles and is supposed to result in the slowest rate of convergence since it takes a relatively long time for information of the best particle to reach the other end of the ring.

Other neighborhood topologies are somewhere in between. Predictably, the effect of different neighborhood topologies on effectiveness and efficiency of the algorithm is problem dependent and is more or less empirically studied.

2.2.6 Biases

It is shown that many metaheuristic optimization algorithms, including PSO, are biased toward some specific regions of the search space. For example, they perform best when the optimum is located at or near the center of the initialization region, which is often the origin [31]. This is particularly true when the information from different members of the population is combined using some sort of averaging operator [32]. Since many of the benchmark optimization problems have their global optimal solutions at or near the origin, such a biased behavior can make the performance evaluation of the algorithms problematic. Different approaches have been taken in order to expose and probably alleviate this bias while testing PSO. Angeline [32] popularized a method called region scaling initially proposed by Gehlhaar and Fogel [33]. The method tries to let the origin outside the initial region covered by the particles by generating the initial solutions in a portion of the search space that does not include origin. Monson and Seppi [31] showed that origin-seeking biases depend on the way that the positions of the particles are updated and region scaling method could not be sufficient for all motion rules. They introduced a center offset method in which the center of the benchmark function under consideration was moved to a different location of the search space. Suganthan et al. [34] also recommended the use of non-biased shifted or rotated benchmark problems.

Clerc [35] showed that this biased behavior can be attributed to the confinement method used, i.e., the method by which the particles are prevented from leaving the search space. A hybrid confinement method is introduced and claimed to be useful in terms of reducing the bias.

Attempts have also been made to propose improved non-biased variants (e.g., Wilke et al. [36]). This is however of less generality in case of unbiased performance comparison because it does not have any effect on the other existing algorithms.

2.3 Hybrid Algorithms

A popular way of producing new improved algorithms is to hybridize two or more existing ones in an attempt to combine favorable features while omitting undesirable aspects. Some of the best results for the real-life and classical optimization problems are obtained using hybrid methods [37]. Numerous different hybrid algorithms using PSO as the main or the supplementary ingredient have been proposed usually in the context of some specific application domain for which that hybrid is particularly well suited [38]. A selection of these methods and approaches is briefly mentioned here along with some examples.

Hybridizing PSO with other metaheuristic algorithms seems to be one of the most popular strategies. This is mainly because the resulting algorithm maintains positive characteristics of metaheuristic algorithms such as global search capability, little dependence on starting point, no need to gradient information, and applicability to non-smooth or non-convex domains. The other metaheuristic algorithm (s) to be hybridized with PSO can be either single agent or population based.

Simulated annealing (SA) [39] is a single-agent metaheuristic algorithm that has been successfully hybridized with PSO. It has been shown in the literature [40] that SA algorithms, when subject to very low variations of temperature parameters and when the solution search for each temperature can reach an equilibrium condition, have very high chances of finding the global optimal solution. Moreover, the metropolis process in SA provides an ability of jumping away from a local optimum. However, SA algorithms require very slow temperature variations and thus increase the required computational effort. On the other hand, although PSO exhibits relatively fast convergence rate, is easy to implement, and is able to find local optimal solutions in a reasonable amount of time, it is notorious of premature convergence, i.e., getting trapped in local optima. Therefore, combining these two algorithms in a judicious way will probably result in a hybridized algorithm with improved performance [41]. Execution of PSO and SA algorithms can be either alternative or sequential. In an alternative execution, every member of the PSO swarm can be considered as an SA single agent at the end of each iteration. Instead, in a sequential execution, the final local solution found by PSO could be considered as a starting point for SA.

As another single-agent metaheuristic algorithm, tabu search (TS) algorithm [42, 43] can have the same effect as SA in hybridization with PSO. The global search could be left to PSO, while TS attempts to improve the suboptimal solutions found by PSO in a local search process. In these hybridized algorithms, TS alleviates premature convergence of PSO while PSO alleviates excessive required computational effort of TS [44].

Hybridization of PSO with other population-based metaheuristic algorithms is more popular. In this case hybridization might signify different meanings. In some hybridized schemes, some techniques are simply borrowed from other algorithms. For example, Løvebjerg et al. [45] borrowed the breeding technique from GAs, i.e., along with standard PSO updating rules, pairs of particles could be chosen to breed

with each other and produce offsprings. Moreover, to keep away from suboptimal solutions, subpopulations were introduced.

Another approach to be mentioned is to use different metaheuristics simultaneously. Krink and Løvebjerg [46] introduced a lifecycle model that allowed for use of PSO, GA, or hill climber by each particle depending on the particle's own preference based on its memory of the best recent improvements. Kaveh and Talatahari [47] introduced a hybridized HPSACO algorithm in which particle swarm optimizer with passive congregation (PSOPC) was used to perform global search task, while ant colony optimization (ACO) [48] was utilized for updating positions of particles to attain the feasible solution space, and harmony search (HS) [49] algorithm was employed for dealing with variable constraints.

In the abovementioned approaches, the position updating rules of the original algorithms need not to be changed. The algorithms are merely operating in combination to each other. Another hybridization approach, however, could be based on combining the updating rules. Higashi and Iba [50] combined GA's Gaussian mutation with velocity and position updating rules of PSO. Juang [51] incorporated mutation, crossover, and elitism. As another example, Kaveh and Talatahari [52] introduced some of the positive aspects of PSO like directing the agents toward the global best and the local best positions into charged system search (CSS) [53] algorithm to improve its performance.

PSO could also be hybridized with techniques and tools other than metaheuristic algorithms. Liu and Abraham [54] hybridized a turbulent PSO with a fuzzy logic controller to produce a fuzzy adaptive TPSO (FATPSO). The fuzzy logic controller was used for adaptively tuning the velocity parameters during an optimization in order to balance exploration and exploitation tendencies. Zahara et al. [55] hybridized Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems. A hybrid PSO-simplex method was also used for damage identification of delaminated beams by Qian et al. [56].

2.4 Discrete PSO

Though PSO has been introduced and more commonly utilized for continuous optimization problems, it can be equally applied to discrete search spaces. A simple and frequently used method to use PSO in discrete problems is to transform the real-valued vectors found by a continuous PSO algorithm into discrete ones. To do this the nearest permitted discrete values could be replaced with any value selected by agents, i.e., a rounding function could be used [57]. However, many discrete and binary PSO variants have been developed that work in discrete search space directly.

The first discrete binary version of PSO is developed by Kennedy and Eberhart [58]. They kept the particle position updating rule unchanged and replaced the velocity in each vector by the probability of a bit in position vector taking the value 1. In other words, if, for example, $v_{i,j} = 0.20$, then there is a twenty percent chance

that $x_{i,j}$ will be a one and an eighty percent chance it will be a zero. In order to keep $v_{i,j}$ in interval [0,1], a sigmoid transformation function was used.

More recently, Chen et al. [59] have proposed a set-based PSO for discrete optimization problems. They have replaced the candidate solutions and velocity vectors by crisp sets and sets with possibilities, respectively. The arithmetic operators in position updating rules are replaced by the operators and procedures defined on such sets.

2.5 Democratic PSO for Structural Optimization

2.5.1 Description of the Democratic PSO

As discussed earlier, different updating strategies have been proposed for PSO resulting in many different variants. Mendes et al. [29] have proposed a fully informed PSO, for example, in which each particle uses the information from all of the other particles in its neighborhood instead of just the best one. It has been shown that the fully informed PSO outperforms the canonical version in all of the mathematical functions under consideration. In a conceptually similar work, Kaveh and Zolghadr [28] have proposed a democratic PSO for structural optimization problems with frequency constraints. Here a brief description of the democratic algorithm is presented as an improved PSO version in the field of structural optimization. The structural optimization under consideration is then introduced in the following section, and the results are then compared to those of the canonical PSO on the same problems reported by Gomes [60].

As indicated before, canonical PSO is notorious for premature convergence, and this can be interpreted as a lack of proper exploration capability. In fact in the standard PSO, all of the particles are just being eagerly attracted toward better solutions. And by each particle, moving toward the best position experienced by itself and by the whole swarm so far is thought of as the only possible way of improvement. Naturally, such an enthusiasm for choosing the shortest possible ways to accomplishment results in some of the better regions of the search space being disregarded.

In a sense, it can be said that the particles of the canonical PSO are only motivated by selfishness (their own preference) and tyranny (the best particle's dictation). Except for their own knowledge and that of the best particle so far, they do not take the achievements of the other members of the swarm into account, i.e., the information is not appropriately shared between the members of the swarm.

In order to address this problem, the velocity vector of the democratic PSO is defined as:

$$v_{i,j}^{k+1} = \chi \left[\omega v_{i,j}^{k} + c_1 r_1 \left(xlbest_{i,j}^{k} - x_{i,j}^{k} \right) + c_2 r_2 \left(xgbest_{j}^{k} - x_{i,j}^{k} \right) + c_3 r_3 d^{k}{}_{i,j} \right] \quad (2.6)$$

in which $d_{i,j}^{k}$ is the jth variable of the vector D for the ith particle. The vector D represents the democratic effect of the other particles of the swarm on the movement of the ith particle. r_3 is a random number uniformly distributed in the range (1,0). Parameter c_3 is introduced to control the weight of the democratic vector. Here, the vector D is taken as:

$$D_i = \sum_{k=1}^{n} Q_{ik} \left(X_k - X_i \right) \quad (2.7)$$

where Q_{ik} is the weight of the kth particle in the democratic movement vector of the ith particle and can be defined as:

$$Q_{ik} = \frac{E_{ik} \frac{obj_{best}}{obj(k)}}{\sum_{j=1}^{n} E_{ij} \frac{obj_{best}}{obj(j)}} \quad (2.8)$$

in which obj stands for objective function value; obj_{best} is the value of the objective function for the best particle in the current iteration; X is the particle's position vector; and E is the eligibility parameter and is analogous to parameter P in CSS [53]. In a minimization problem, E can be defined as:

$$\mathrm{E_{ik}} = \begin{cases} 1 & \frac{obj(k) - obj(i)}{obj_{worst} - obj_{best}} > rand \vee obj(k) < obj(i) \\ 0 & else \end{cases} \quad (2.9)$$

where obj_{worst} and obj_{best} are the values of the objective function for the worst and the best particles in the current iteration, respectively. The symbol $\vee$ stands for union. Schematic movement of a particle is illustrated in Fig. 2.3.

Since a term is added to the velocity vector of PSO, the parameter χ should be decreased in order to avoid divergence. Here, this parameter is determined using a trial and error process. It seems that a value in the range (0.4, 0.5) is suitable for the problems under consideration.

As it can be seen, the democratic PSO makes use of the information produced by all of the eligible members of the swarm in order to determine the new position of each particle. In fact, according to Eq. (2.9), all of the better particles and some of the worse particles affect the new position of the particle under consideration. This modification enhances the performance of the algorithm in two ways: (1) helping the agents to receive information about good regions of the search space other than those experienced by themselves and the best particle of the swarm and (2) letting

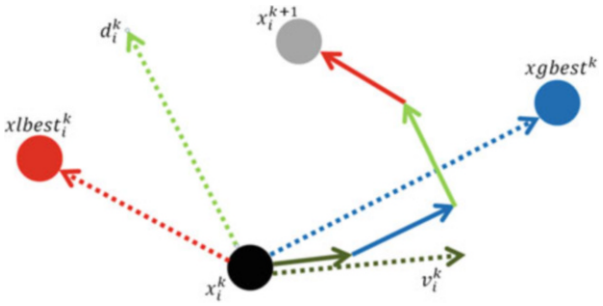

Fig. 2.3 Schematic movement of a particle based on Eq. (2.6)

some bad particles take part in the movement of the swarm and thus improving the exploration capabilities of the algorithm. Both of the above effects help to alleviate the premature convergence of the algorithm.

Numerical results show that this simple modification which does not call for any extra computational effort meaningfully enhances the performance of the PSO.

2.5.2 *Truss Layout and Size Optimization with Frequency Constraints*

In a frequency constraint truss layout and size optimization problem, the aim is to minimize the weight of the structure while satisfying some constraints on natural frequencies. The design variables are considered to be the cross-sectional areas of the members and/or the coordinates of some nodes. The topology of the structure is not supposed to be changed, and thus the connectivity information is predefined and kept unaffected during the optimization process. Each of the design variables should be chosen within a permissible range. The optimization problem can be stated mathematically as follows:

$$\begin{aligned} &\textit{Find } X = [x_1, x_2, x_3, \ldots, x_n] \\ &\textit{to minimizes } P(X) = f(X) \times f_{penalty}(X) \\ &\textit{subjected to} \\ &\omega_j \leq \omega_j^* \quad \text{for some natural frequencies } j \\ &\omega_k \geq \omega_k^* \quad \text{for some natural frequencies } k \\ &x_{imin} \leq x_i \leq x_{imax} \end{aligned} \tag{2.10}$$

where X is the vector of the design variables, including both nodal coordinates and cross-sectional areas; n is the number of variables which is naturally affected by the element grouping scheme which in turn is chosen with respect to the symmetry and practice requirements; $P(X)$ is the penalized cost function or the objective function to be minimized; $f(X)$ is the cost function, which is taken as the weight of the structure in a weight optimization problem; and $f_{penalty}(X)$ is the penalty function which is used to make the problem unconstrained. When some constraints

corresponding to the response of the structure are violated in a particular solution, the penalty function magnifies the weight of the solution by taking values bigger than one; ω_j is the jth natural frequency of the structure and ω_j^* is its upper bound. ω_k is the kth natural frequency of the structure and ω_k^* is its lower bound. x_{imin} and x_{imax} are the lower and upper bounds of the design variable x_i, respectively.

The cost function is expressed as:

$$f(X) = \sum_{i=1}^{nm} \rho_i L_i A_i \tag{2.11}$$

where ρ_i is the material density of member i; L_i is the length of member i; and A_i is the cross-sectional area of member i.

The penalty function is defined as:

$$f_{penalty}(X) = (1 + \varepsilon_1 . v)^{\varepsilon_2}, \quad v = \sum_{i=1}^{q} v_i \tag{2.12}$$

where q is the number of frequency constraints.

$$v_i = \begin{cases} 0 & \textit{if the ith constraint is satisfied} \\ \left| 1 - \dfrac{\omega_i}{\omega^*_i} \right| & \textit{else} \end{cases} \tag{2.13}$$

The parameters ε_1 and ε_2 are selected considering the exploration and the exploitation balance of the problem. In this study, ε_1 is taken as unity, and ε_2 starts from 1.5 and linearly increases to 6 in all test examples. These values penalize the unfeasible solutions more severely as the optimization process proceeds. As a result, in the early stages, the agents are free to explore the search space, but at the end they tend to choose solutions without violation.

2.5.3 Numerical Examples

Four numerical examples from the field of truss layout and size optimization are provided in this section in order to examine the viability of the proposed algorithm and to compare it with the canonical PSO to clarify the effect of the modifications. The results are compared with those of the canonical version and some other methods reported in the literature.

Parameter χ is set to 0.5 in all numerical examples while parameter c_3 is set to 4. A total population of 30 particles is utilized for all of the examples. Each example has been solved 30 times independently. In all the examples, the termination criterion is taken as the number of iterations. A total number of 200 iterations are considered for all of the examples. The side constraints are handled using an

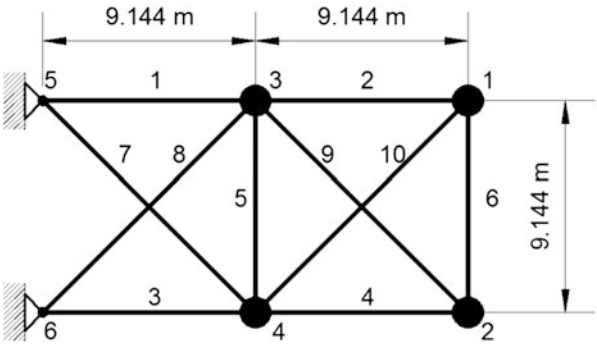

Fig. 2.4 Schematic of the planar 10-bar truss structure

HS-based constraint handling technique, as introduced by Kaveh and Talatahari [47]. Any other appropriate side constraint handling technique might be used.

2.5.3.1 A 10-Bar Truss

For the first example, size optimization of a 10-bar planar is considered. The configuration of the structure is depicted in Fig. 2.4.

This is a well-known benchmark problem in the field of frequency constraint structural optimization. Each of the members' cross-sectional areas is assumed to be an independent variable. A nonstructural mass of 454.0 kg is attached to all free nodes. Table 2.1 summarizes the material properties, variable bounds, and frequency constraints for this example.

This problem has been investigated by different researchers: Grandhi and Venkayya [61] using an optimality algorithm, Sedaghati et al. [62] using a sequential quadratic programming and finite element force method, Wang et al. [63] using an evolutionary node shift method, Lingyun et al. [64] utilizing a niche hybrid genetic algorithm, Gomes employing the standard particle swarm optimization algorithm [60], and Kaveh and Zolghadr [65, 66] utilizing the standard and an enhanced CSS and a hybridized CSS–BBBC with a trap recognition capability.

The design vectors and the corresponding masses of the optimal structures found by different methods are summarized in Table 2.2.

It should be noted that a modulus of elasticity of $E = 6.98 \times 10^{10}$ Pa is used in Gomes [60] and Kaveh and Zolghadr [65]. This will generally result in relatively lighter structures. Considering this, it appears that the proposed algorithm has obtained the best solution so far. Particularly, the optimal structure found by the algorithm is more than 5.59 kg lighter than that of the standard PSO in spite of using smaller value for modulus of elasticity. Using $E = 6.98 \times 10^{10}$ Pa, DPSO finds a structure weighted 524.70 kg which is about 13 kg lighter than that of standard PSO. The mean weight and the standard deviation of the results gained by DPSO are 537.80 kg and 4.02 kg, respectively, while PSO has obtained a mean weight of 540.89 kg and a standard deviation of 6.84 kg. This means that DPSO performs better than the standard PSO in terms of best weight, average weight, and standard deviation.

Table 2.1 Material properties, variable bounds, and frequency constraints for the 10-bar truss structure

Property/unit	Value
E(modulus of elasticity)/N/m^2	6.89×10^{10}
ρ(material density)/kg/m^3	2770.0
Added mass/kg	454.0
Design variable lower bound/m^2	0.645×10^{-4}
Design variable upper bound/m^2	50×10^{-4}
L(main bar's dimension)/m	9.144
Constraints on the first three frequencies/Hz	$\omega_1 \geq 7$, $\omega_2 \geq 15$, $\omega_3 \geq 20$

Table 2.2 Optimized designs (cm^2) obtained for the planar 10-bar truss problem (the optimized weight does not include the added masses)

Element number	Grandhi and Venkayya [61]	Sedaghati et al. [62]	Wang et al. [63]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65]	Proposed algorithm
						Standard CSS	
1	36.584	38.245	32.456	42.23	37.712	38.811	35.944
2	24.658	9.916	16.577	18.555	9.959	9.0307	15.530
3	36.584	38.619	32.456	38.851	40.265	37.099	35.285
4	24.658	18.232	16.577	11.222	16.788	18.479	15.385
5	4.167	4.419	2.115	4.783	11.576	4.479	0.648
6	2.070	4.419	4.467	4.451	3.955	4.205	4.583
7	27.032	20.097	22.810	21.049	25.308	20.842	23.610
8	27.032	24.097	22.810	20.949	21.613	23.023	23.599
9	10.346	13.890	17.490	10.257	11.576	13.763	13.135
10	10.346	11.452	17.490	14.342	11.186	11.414	12.357
Weight (kg)	594.0	537.01	553.8	542.75	537.98	531.95	532.39

Table 2.3 represents the natural frequencies of the optimized structures obtained by different methods.

Figure 2.5 compares the convergence curves for the 10-bar planar truss obtained by the democratic PSO and the standard PSO.

The termination criterion is not clearly stated in reference [60]. It is just declared that a combination of three different criteria was simultaneously employed: (1) the differences in the global best design variables between two consecutive iterations, (2) the differences of the global best objective function, and (3) the coefficient of variation of objective function in the swarm. In any case, it seems no improvement is expected from PSO after the 2000th analysis, and hence the execution is terminated.

Comparison of the convergence curves above provides some useful points about the differences of the two algorithms. The standard and the democratic PSO utilize 50 and 30 particles for this problem, respectively. Although the standard PSO uses

Table 2.3 Natural frequencies (Hz) evaluated at the optimized designs for the planar 10-bar truss

Frequency number	Grandhi and Venkayya [61]	Sedaghati et al. [62]	Wang et al. [63]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65]	Proposed algorithm
						Standard CSS	
1	7.059	6.992	7.011	7.008	7.000	7.000	7.000
2	15.895	17.599	17.302	18.148	17.786	17.442	16.187
3	20.425	19.973	20.001	20.000	20.000	20.031	20.000
4	21.528	19.977	20.100	20.508	20.063	20.208	20.021
5	28.978	28.173	30.869	27.797	27.776	28.261	28.470
6	30.189	31.029	32.666	31.281	30.939	31.139	29.243
7	54.286	47.628	48.282	48.304	47.297	47.704	48.769
8	56.546	52.292	52.306	53.306	52.286	52.420	51.389

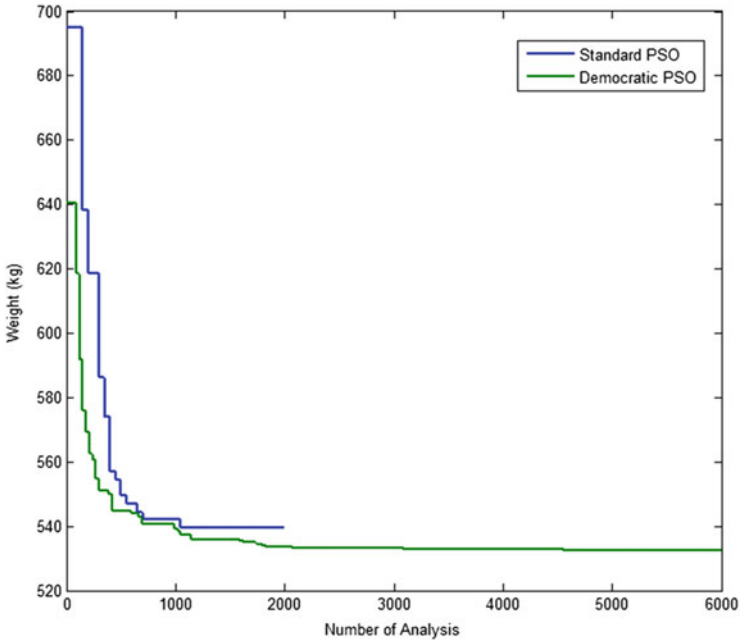

Fig. 2.5 Comparison of convergence curves of democratic and standard PSO algorithms recorded in the 10-bar problem

more particles which is supposed to maintain better coverage of the search space and higher level of exploration, its convergence curve shows that the convergence is almost attained within the first 1000 analyses and after that the convergence curve becomes straight. On the other hand, democratic PSO reaches an initial convergence after about 1500th analyses, and it still keeps exploring the search space until it reaches the final result at 3000th analysis. This can be interpreted as the modifications being effective on the alleviation of the premature convergence problem. It should be noted that the structure found by DPSO at 2000th analysis is much lighter than that found by PSO at the same analysis. In fact while the modifications improve the exploration capabilities of the algorithm, they do not disturb the algorithm's convergence task.

2.5.3.2 A Simply Supported 37-Bar Planar Truss

A simply supported 37-bar Pratt type truss, as depicted in Fig. 2.6, is examined as the second example.

The elements of the lower chord are modeled as bar elements with constant rectangular cross-sectional areas of 4×10^{-3} m^2. The other members are modeled as bar elements. These members which form the sizing variables of the problem are grouped with respect to symmetry. Also, the y-coordinate of all the nodes on the upper chord can vary in a symmetrical manner to form the layout variables. On the

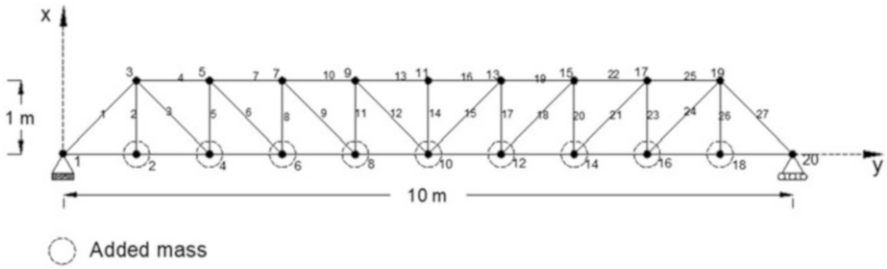

Fig. 2.6 Schematic of the simply supported planar 37-bar truss

Table 2.4 Material properties and frequency constraints for the simply supported planar 37-bar truss

Property/unit	Value
E(modulus of elasticity)/N/m^2	2.1×10^{11}
ρ(material density)/kg/m^3	7800
Design variable lower bound/m^2	1×10^{-4}
Design variable upper bound/m^2	10×10^{-4}
Added mass/kg	10
Constraints on first three frequencies/Hz	$\omega 1 \geq 20$, $\omega 2 \geq 40$, $\omega 3 \geq 60$

lower chord, a nonstructural mass of 10 kg is attached to all free nodes. The first three natural frequencies of the structure are considered as the constraints. So this is an optimization on layout and size with 19 design variables (14 sizing variables + five layout variables) and three frequency constraints. This example has been studied by Wang et al. [63] using an evolutionary node shift method and Lingyun et al. [64] using a niche hybrid genetic algorithm. Gomes [60] has investigated this problem using the standard particle swarm algorithm. Kaveh and Zolghadr [65] used the standard and an enhanced CSS to optimize the structure.

Material properties, frequency constrains, and added masses are listed in Table 2.4.

Final cross-sectional areas and node coordinates obtained by different methods together with the corresponding weight are presented in Table 2.5. It can be seen that the proposed algorithm has found the best results so far. Specifically, in comparison to the standard PSO, the resulted structure is meaningfully lighter.

The mean weight and the standard deviation of the results obtained by DPSO are 362.21 kg and 1.68 kg, respectively, while PSO has obtained a mean weight of 381.2 kg and a standard deviation of 4.26 kg. This indicates that DPSO not only finds a better best solution but also is more stable.

Table 2.6 represents the natural frequencies of the final structures obtained by various methods for the 37-bar simply supported planar truss.

Figure 2.7 shows the optimized layout of the simply supported 37-bar truss as found by DPSO. The convergence curves for the democratic PSO and the standard PSO are shown in Fig. 2.6. The information on the convergence curve values at the few first analyses is not available in [60] (Fig. 2.8).

Table 2.5 Optimized designs obtained for the planar 37-bar truss problem

Variable	Wang et al. [63]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65] Standard CSS	Proposed algorithm
Y3, Y19 (m)	1.2086	1.1998	0.9637	0.8726	0.9482
Y5, Y17 (m)	1.5788	1.6553	1.3978	1.2129	1.3439
Y7, Y15 (m)	1.6719	1.9652	1.5929	1.3826	1.5043
Y9, Y13 (m)	1.7703	2.0737	1.8812	1.4706	1.6350
Y11 (m)	1.8502	2.3050	2.0856	1.5683	1.7182
A1, A27 (cm^2)	3.2508	2.8932	2.6797	2.9082	2.6208
A2, A26 (cm^2)	1.2364	1.1201	1.1568	1.0212	1.0397
A3, A24 (cm^2)	1.0000	1.0000	2.3476	1.0363	1.0464
A4, A25 (cm^2)	2.5386	1.8655	1.7182	3.9147	2.7163
A5, A23 (cm^2)	1.3714	1.5962	1.2751	1.0025	1.0252
A6, A21 (cm^2)	1.3681	1.2642	1.4819	1.2167	1.5081
A7, A22 (cm^2)	2.4290	1.8254	4.6850	2.7146	2.3750
A8, A20 (cm^2)	1.6522	2.0009	1.1246	1.2663	1.4498
A9, A18 (cm^2)	1.8257	1.9526	2.1214	1.8006	1.4499
A10, A19 (cm^2)	2.3022	1.9705	3.8600	4.0274	2.5327
A11, A17 (cm^2)	1.3103	1.8294	2.9817	1.3364	1.2358
A12, A15 (cm^2)	1.4067	1.2358	1.2021	1.0548	1.3528
A13, A16 (cm^2)	2.1896	1.4049	1.2563	2.8116	2.9144
A14 (cm^2)	1.0000	1.0000	3.3276	1.1702	1.0085
Weight (kg)	366.50	368.84	377.20	362.84	360.40

Table 2.6 Natural frequencies (Hz) evaluated at the optimized designs for the planar 37-bar truss

Frequency number	Wang et al. [63]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65] Standard CSS	Proposed algorithm
1	20.0850	20.0013	20.0001	20.0000	20.0194
2	42.0743	40.0305	40.0003	40.0693	40.0113
3	62.9383	60.0000	60.0001	60.6982	60.0082
4	74.4539	73.0444	73.0440	75.7339	76.9896
5	90.0576	89.8244	89.8240	97.6137	97.2222

2.5.3.3 A 52-Bar Dome-Like Truss

Simultaneous layout and size optimization of a 52-bar dome-like truss is considered as the third example. Initial layout of the structure is depicted in Fig. 2.9. Nonstructural masses of 50 kg are attached to all free nodes.

Table 2.7 summarized the material properties, frequency constraints, and variable bounds for this example.

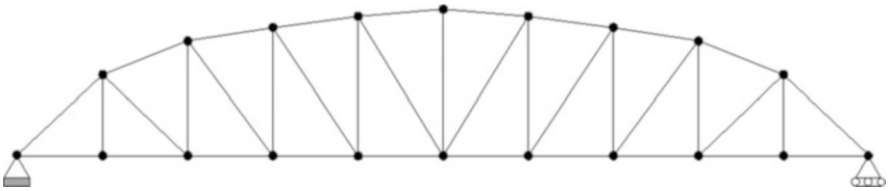

Fig. 2.7 Schematic of the optimized layout of the simply supported planar 37-bar truss

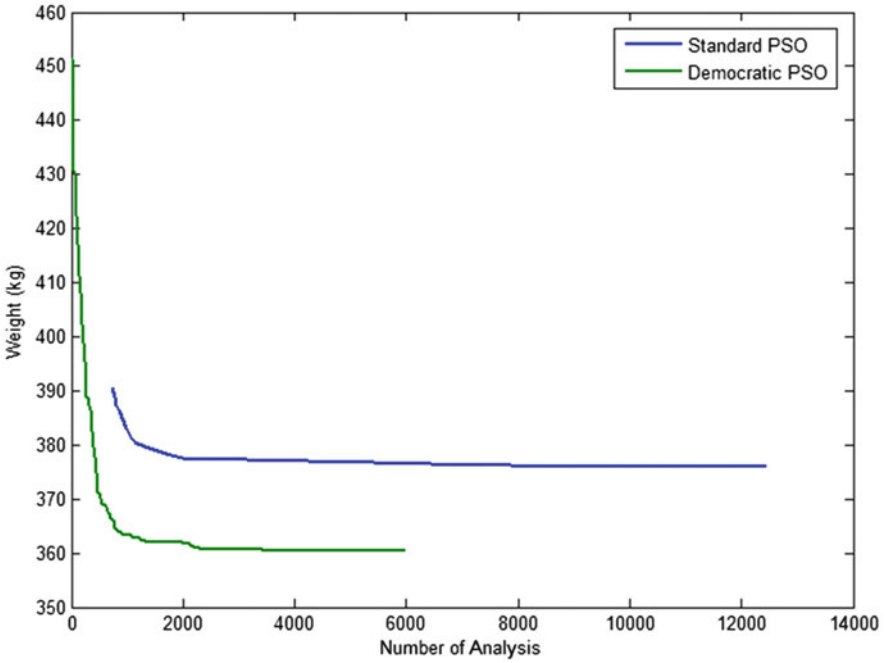

Fig. 2.8 Comparison of convergence curves of democratic and standard PSO algorithms recorded in the 37-bar Pratt type planar truss

All of the elements of the structure are categorized in 8 groups according to Table 2.8. All free nodes are permitted to move ±2 m from their initial position in a symmetrical manner. This is a configuration optimization problem with 13 variables (eight sizing variables + five layout variables) and two frequency constraints.

This example has been investigated by Lin et al. [67] using a mathematical programming technique and Lingyun et al. [64] using a niche hybrid genetic algorithm. Gomes [60] has analyzed this problem using the standard particle swarm algorithm. The authors have studied the problem using the standard and an enhanced CSS [65] and a hybridized CSS–BBBC with a trap recognition capability [66].

Table 2.9 compares the final cross-sectional areas and node coordinates found by different methods together with the corresponding weight for the 52-bar space truss.

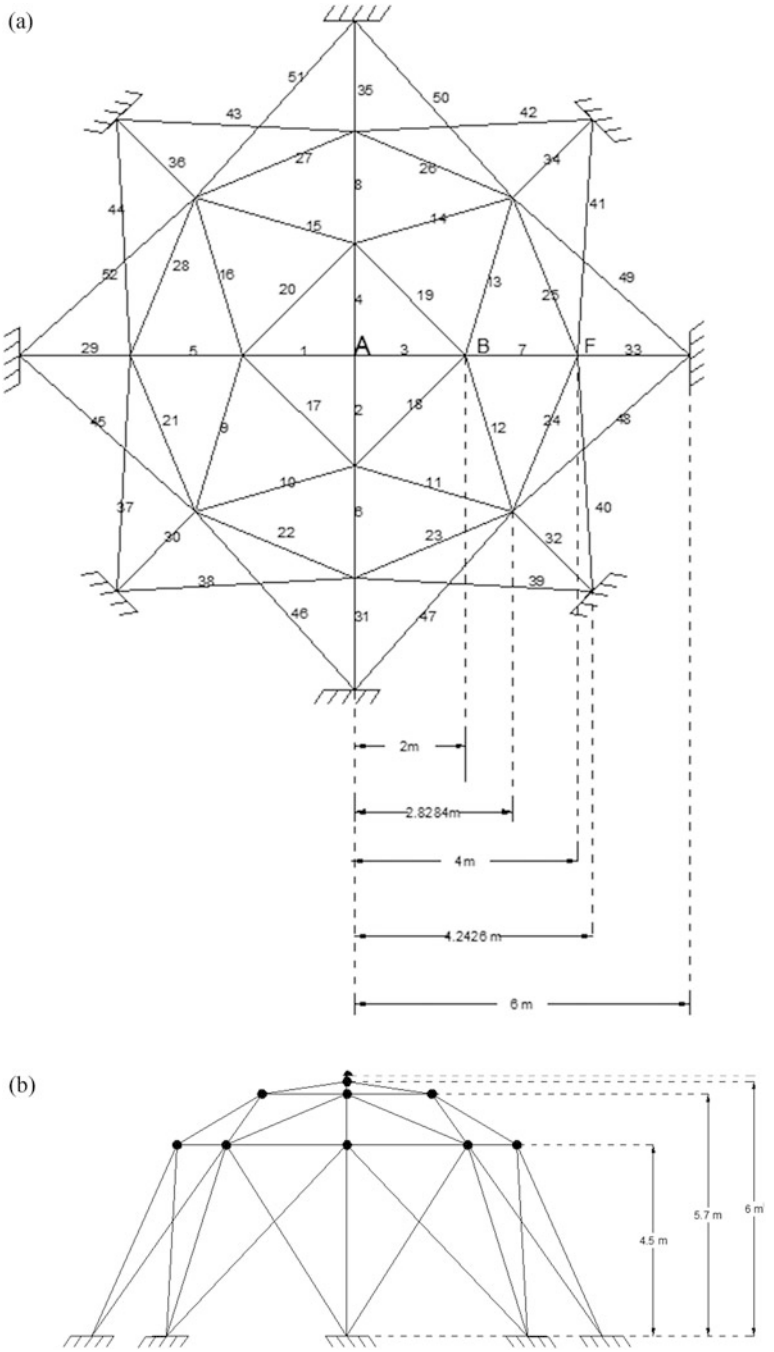

Fig. 2.9 Schematic of the initial layout of the spatial 52-bar truss. (**a**) Top view, (**b**) side view

Table 2.7 Material properties and frequency constraints and variable bounds for the spatial 52-bar truss

Property/unit	Value
E(modulus of elasticity)/N/m^2	2.1×10^{11}
ρ(material density)/kg/m^3	7800
Added mass/kg	50
Allowable range for cross sections/m^2	$0.0001 \leq A \leq 0.001$
Constraints on the first three frequencies/Hz	$\omega_1 \leq 15.916$ $\omega_2 \geq 28.648$

Table 2.8 Element grouping adopted in the spatial 52-bar truss problem

Group number	Elements
1	1–4
2	5–8
3	9–16
4	17–20
5	21–28
6	29–36
7	37–44
8	45–52

Table 2.9 Optimized designs obtained for the spatial 52-bar truss problem

Variable	Lin et al. [67]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65] Standard CSS	Present work
Z_A (m)	4.3201	5.8851	5.5344	5.2716	6.1123
X_B (m)	1.3153	1.7623	2.0885	1.5909	2.2343
Z_B (m)	4.1740	4.4091	3.9283	3.7093	3.8321
X_F (m)	2.9169	3.4406	4.0255	3.5595	4.0316
Z_F (m)	3.2676	3.1874	2.4575	2.5757	2.5036
A1 (cm^2)	1.00	1.0000	0.3696	1.0464	1.0001
A2 (cm^2)	1.33	2.1417	4.1912	1.7295	1.1397
A3 (cm^2)	1.58	1.4858	1.5123	1.6507	1.2263
A4 (cm^2)	1.00	1.4018	1.5620	1.5059	1.3335
A5 (cm^2)	1.71	1.911	1.9154	1.7210	1.4161
A6 (cm^2)	1.54	1.0109	1.1315	1.0020	1.0001
A7 (cm^2)	2.65	1.4693	1.8233	1.7415	1.5750
A8 (cm^2)	2.87	2.1411	1.0904	1.2555	1.4357
Weight (kg)	298.0	236.046	228.381	205.237	195.351

It can be seen that the result gained by the democratic PSO is far better than the standard PSO. The standard PSO uses 70 particles and about 160 iterations (11,200 analyses) to reach its best result, while the democratic PSO uses 30 particles and 200 iterations (6000 analyses). Table 2.8 indicates that among all the methods listed above, the democratic PSO has obtained the best solution. The mean weight and the

Table 2.10 Natural frequencies (Hz) evaluated at the optimized designs for the spatial 52-bar truss

Frequency number	Lin et al. [67]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65]	Present work
				Standard CSS	
1	15.22	12.81	12.751	9.246	11.315
2	29.28	28.65	28.649	28.648	28.648
3	29.28	28.65	28.649	28.699	28.648
4	31.68	29.54	28.803	28.735	28.650
5	33.15	30.24	29.230	29.223	28.688

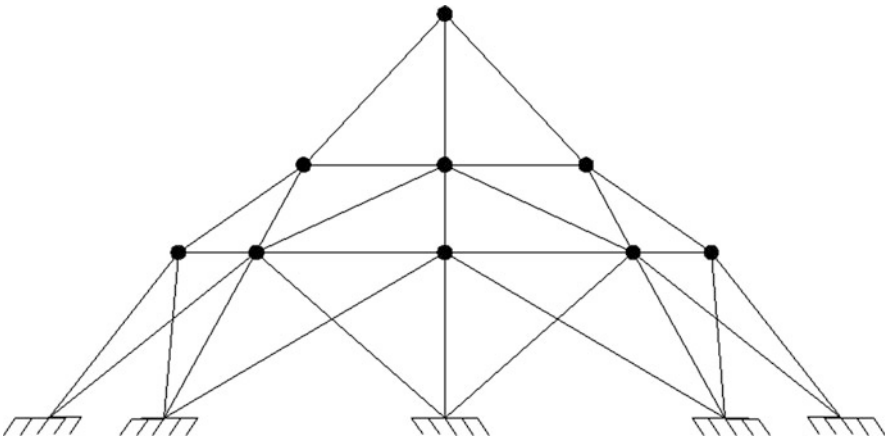

Fig. 2.10 Schematic of the optimized layout of the spatial 52-bar truss

standard deviation of the results gained by DPSO are 198.71 kg and 13.85 kg, respectively, while PSO has obtained a mean weight of 234.3 kg and a standard deviation of 5.22 kg. DPSO performs considerably better in terms of best and mean weight.

Table 2.10 shows the natural frequencies of the final structures found by various methods for the 52-bar dome-like space truss.

Figure 2.10 shows the optimized layout of the spatial 52-bar truss as found by DPSO. The convergence curve of the best run of the democratic PSO for the 52-bar dome-like truss is shown Fig. 2.11. The convergence curve for the standard PSO is not available in reference [60].

2.5.3.4 A 120-Bar Dome Truss

The 120-bar dome truss shown in Fig. 2.12 is considered as the last example. This problem has been previously studied as a benchmark optimization problem with static constraints.

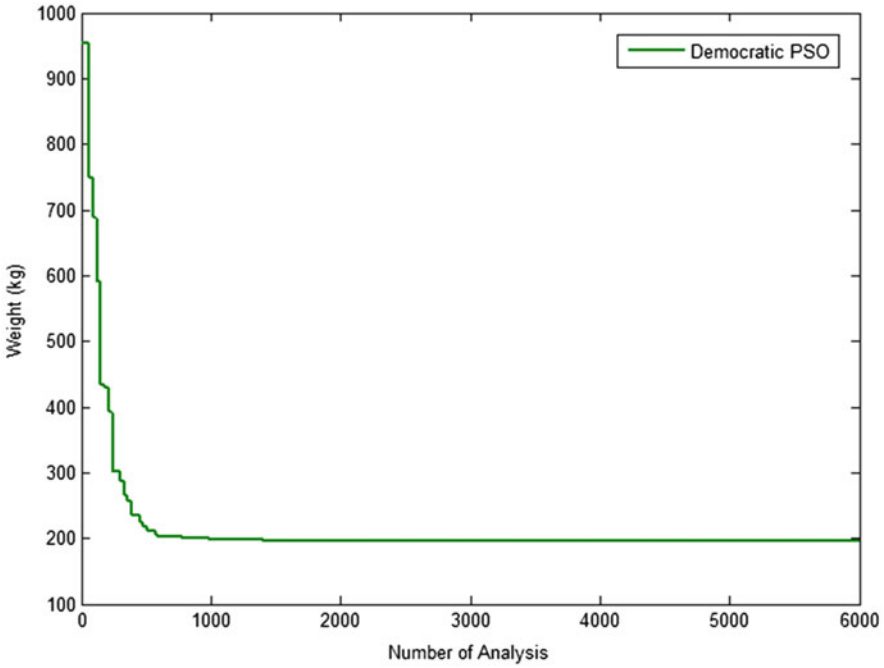

Fig. 2.11 Convergence curve of the democratic PSO for the spatial 52-bar truss

The authors used the problem as a size optimization problem with frequency constraints in [65]. Nonstructural masses are attached to all free nodes as follows: 3000 kg at node one, 500 kg at nodes 2 through 13, and 100 kg at the rest of the nodes. Material properties, frequency constraints, and variable bounds for this example are summarized in Table 2.11. The layout of the structure is kept unchanged during the optimization process. Hence, this is a sizing optimization problem.

This example is solved here using both the standard and democratic PSO in order to make the comparison possible. Thirty particles and 200 iterations are used for both methods. Table 2.12 represents a comparison between the final results obtained by the standard and the democratic PSO. Table 2.13 shows the natural frequencies of the final structures found by both methods.

According to Table 2.12, the result obtained by the democratic PSO is meaningfully lighter than that of the standard PSO. The mean weight and the standard deviation of the results gained by DPSO are 8895.99 kg and 4.26 kg, respectively, while PSO has obtained a mean weight of 9251.84 kg and a standard deviation of 89.38 kg. This shows that the democratic PSO outperforms the standard version in all of the abovementioned aspects. Figure 2.13 shows the convergence curves for both methods.

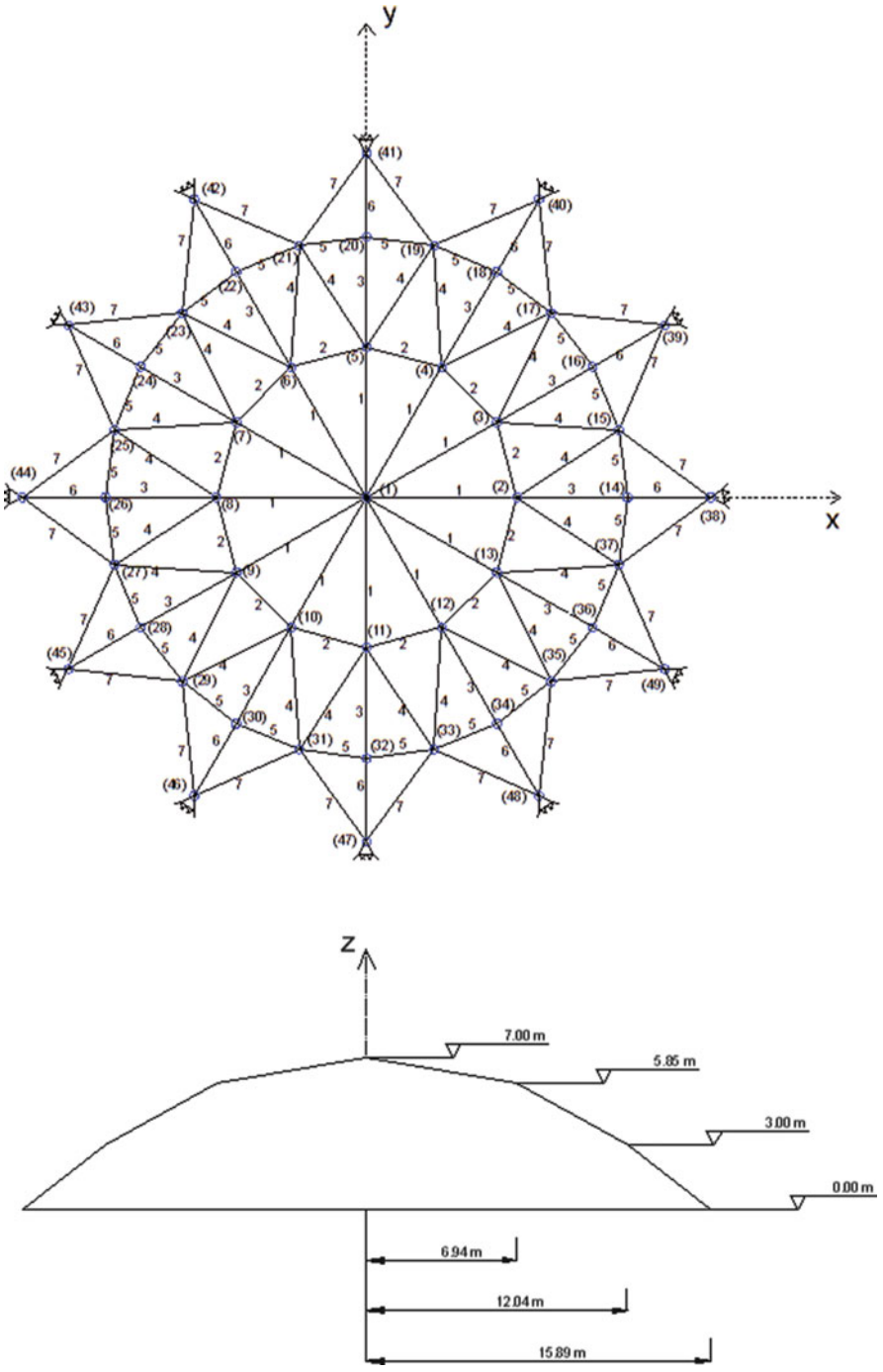

Fig. 2.12 Schematic of the 120 bar

Table 2.11 Material properties and frequency constraints and variable bounds for the 120-bar dome truss

Property/unit	Value
E(Modulus of elasticity)/N/m^2	2.1×10^{11}
ρ(Material density)/kg/m^3	7971.810
Added mass/kg	$m_1 = 3000$, $m_1 = 500$, $m_2 = 100$
Allowable range for cross sections/m^2	$0.0001 \leq A \leq 0.01293$
Constraints on first three frequencies/Hz	$\omega_1 \geq 9$ $\omega_2 \geq 11$

Table 2.12 Optimized designs (cm^2) obtained for the 120-bar dome truss

Element group	Standard PSO	Democratic PSO
1	23.494	19.607
2	32.976	41.290
3	11.492	11.136
4	24.839	21.025
5	9.964	10.060
6	12.039	12.758
7	14.249	15.414
Weight (kg)	9171.93	8890.48

Table 2.13 Natural frequencies (Hz) evaluated at the optimized designs for the 120-bar dome truss

Frequency number	Standard PSO	Democratic PSO
1	9.0000	9.0001
2	11.0000	11.0007
3	11.0052	11.0053
4	11.0134	11.0129
5	11.0428	11.0471

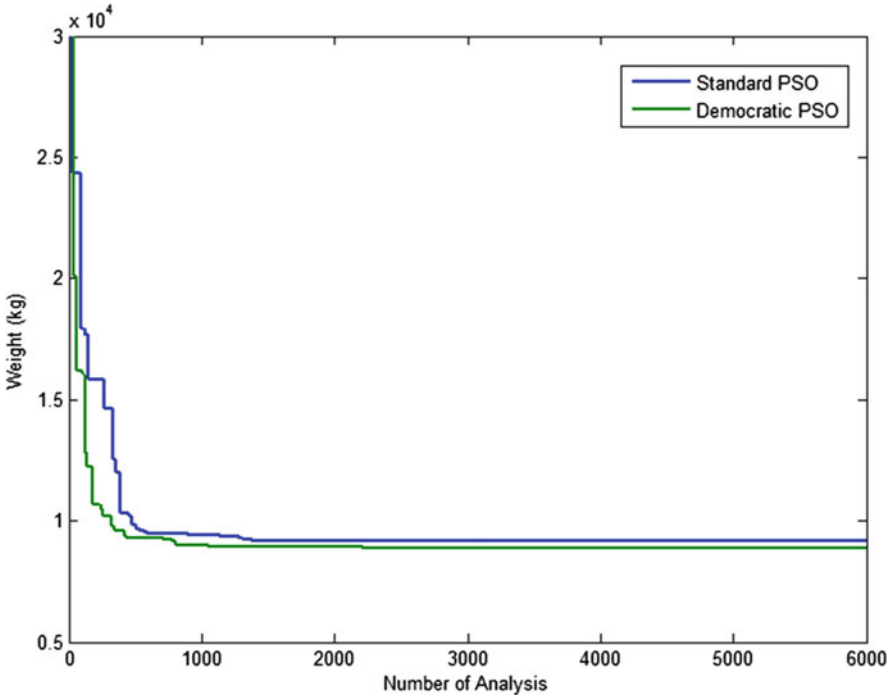

Fig. 2.13 Comparison of converge curves of democratic and standard PSO algorithms recorded in the 120-bar dome problem

References

1. Kennedy J, Eberhart R (1995) Particle swarm optimization. Proc IEEE Int Conf Neural Netw 4:1942–1948
2. Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: Proceedings of IEEE World Congress on computational intelligence. The 1998 I.E. international conference on evolutionary computation, pp 69–73
3. Reeves WT (1983) Particle systems—a technique for modeling a class of fuzzy objects. ACM Trans Graph 2(2):91–108
4. Renolds CW (1987) Flocks, herds, and schools: a distributed behavioral model. Comput Graph 21(4):25–34 (Proc SIGGRAPH'87)
5. Millonas MM (1993) Swarms, phase transitions, and collective intelligence. In: Langton CG (ed) Proceedings of ALIFE III. Santa Fe Institute, Addison-Wesley, USA
6. Heppner F, Grenander U (1990) A stochastic nonlinear model for coordinated bird flocks. In: Krasner S (ed) The ubiquity of chaos. AAAS Publications, Washington, DC
7. Eberhart RC, Simpson P, Dobbins R (1996) Computational intelligence PC tools. AP Professional, San Diego, CA, pp 212–226, Chapter 6
8. Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. In: Proceedings of the congress on evolutionary computation, pp 73–79

9. Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of IEEE congress evolutionary computation, San Diego, CA, pp 84–88
10. Clerc M (1999) The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: Proceedings of I999 ICEC, Washington, DC, pp 1951–1957
11. Bui LT, Soliman O, Abass HS (2007) A modified strategy for the constriction factor in particle swarm optimization. In: Randall M, Abass HS, Wiles J (eds) Lecture notes in artificial intelligence 4828. pp 333–344
12. Kennedy J (2006) Swarm intelligence. In: Handbook of nature-inspired and innovative computing. Springer, New York, pp 187–219
13. Talbi EG (2009) Metaheuristics: from design to implementation. Wiley, UK
14. Shi Y, Eberhart RC (1998) Parameter selection in particle swarm optimization. In: The proceedings of evolutionary programming VII (EP98), pp 591–600
15. Carlisle A, Dozier G (2001) An off-the-shelf PSO. In: Proceedings of workshop on particle swarm optimization, Indianapolis, IN
16. Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf Process Lett 85:317–325
17. Zhang L, Yu H, Hu S (2005) Optimal choice of parameters for particle swarm optimization. J Zhejiang Univ Sci 6A(6):528–534
18. Pedersen MEH (2010) Good parameters for particle swarm optimization. Hvass Laboratories Technical Report HL1001
19. Bansal JC, Singh PK, Saraswat M, Verma A, Jadon SS, Abraham A (2011) Inertia weight strategies in particle swarm optimization. In: Third world congress on nature and biologically inspired computing (NaBIC 2011), IEEE, Salamanca, Spain, pp 640–647
20. Wang Y, Li B, Weise T, Wang J, Yuan B, Tian Q (2011) Self-adaptive learning based particle swarm optimization. Inform Sci 181(20):4515–4538
21. Angeline PJ (1998) Evolutionary optimization versus particle swarm optimization: philosophy and performance difference. In: Proceedings of 7th annual conference on evolutionary programming, p 601
22. Zhao Y, Zub W, Zeng H (2009) A modified particle swarm optimization via particle visual modeling analysis. Comput Math Appl 57:2022–2029
23. van den Bergh F, Engelbrecht AP (2002) A new locally convergent particle swarm optimizer. In: Proceedings of IEEE conference on systems, man and cybernetics, Hammamet, Tunisia
24. Krink T, Vestertroem JS, Riget J (2002) Particle swarm optimization with spatial particle extension. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2002), Honolulu, Hawaii
25. Riget J, Vesterstrøm JS (2002) A diversity-guided particle swarm optimizer—the ARPSO. EVALife Technical Report No 2002–2002
26. Silva A, Neves A, Costa E (2002) An empirical comparison of particle swarm and predator prey optimization. In: Proceedings of 13th Irish international conference on artificial intelligence and cognitive science 2464, pp 103–110
27. Jie J, Zeng J, Han CZ (2006) Adaptive particle swarm optimization with feedback control of diversity. In: Proceedings of the 2006 international conference on computational intelligence and bioinformatics (ICIC'06)—Volume Part III, pp 81–92
28. Kaveh A, Zolghadr A (2013) A democratic PSO for truss layout and size optimization with frequency constraints. Comput Struct 42(3):10–21
29. Mendes R, Kennedy J, Neves J (2004) The fully informed particle swarm: simpler, maybe better. IEEE Trans Evolut Comput 8(3):204–210
30. Matsushita H, Nishio Y (2009) Network-structured particle swarm optimizer with various topology and its behaviors. In: Advances in self-organizing maps, Lecture notes in computer science 5629. pp 163–171

31. Monson CK, Seppi KD (2005) Exposing origin-seeking bias in PSO. In: Proceedings of the conference on genetic and evolutionary computation (GECCO'05), Washington DC, USA, pp 241–248
32. Angeline PJ (1998) Using selection to improve particle swarm optimization. In: Proceedings of the IEEE congress on evolutionary computation (CEC 1998), Anchorage, Alaska, USA
33. Gehlhaar DK, Fogel DB (1996) Tuning evolutionary programming for conformationally flexible molecular docking. In: Evolutionary programming, pp 419–429
34. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization. Nanyang Technological University, Singapore
35. Clerc M (2006) Confinements and biases in particle swarm optimisation. Open access archive HAL
36. Wilke DN, Kok S, Groenwold AA (2007) Comparison of linear and classical velocity update rules in particle swarm optimization: notes on scale and frame invariance. Int J Numer Methods Eng 70:985–1008
37. Talbi E-G (2002) A taxonomy of hybrid metaheuristics. J Heuristics 8:541–564
38. Banks A, Vincent J, Anyakoha C (2008) A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Nat Comput 7(1):109–124
39. Černý V (1985) Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. J Optim Theory Appl 45:41–51
40. Locatelli M (1996) Convergence properties of simulated annealing for continuous global optimization. J Appl Probab 33:1127–1140
41. Shieh HL, Kuo CC, Chiang CM (2011) Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification. Appl Math Comput 218:4365–4383
42. Glover F (1989) Tabu search—part 1. ORSA J Comput 1(2):190–206
43. Glover F (1990) Tabu search—part 2. ORSA J Comput 2(1):4–32
44. Shen Q, Shi WM, Kong W (2008) Hybrid particle swarm optimization and tabu search approach for selecting genes for tumor classification using gene expression data. Comput Biol Chem 32:53–60
45. Løvbjerg M, Rasmussen TK, Krink T (2001) Hybrid particle swarm optimizer with breeding and subpopulations In: Proceedings of the genetic and evolutionary computation conference (GECCO-2001)
46. Krink T, Løvbjerg M (2002) The lifecycle model: combining particle swarm optimization, genetic algorithms and hillclimbers. In: Proceedings of parallel problem solving from nature VII (PPSN 2002). Lecture notes in computer science (LNCS) No 2439, pp 621–630
47. Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87(56):267–283
48. Dorigo M (1992) Optimization, learning and natural algorithms (in Italian). PhD Thesis, Dipartimento di Elettronica, Politecnico di Milano, IT
49. Geem ZW, Kim J-H, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
50. Higashi N, Iba H (2003) Particle swarm optimization with Gaussian mutation. In: Proceedings of the IEEE swarm intelligence symposium 2003 (SIS 2003), Indianapolis, Indiana, USA, pp 72–79
51. Juang C-F (2004) A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans Syst Man Cybern Part B Cybern 34(2):997–1006
52. Kaveh A, Talatahari S (2011) Hybrid charged system search and particle swarm optimization for engineering design problems. Eng Comput 28(4):423–440
53. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213(3–4):267–289

54. Liu H, Abraham A (2005) Fuzzy adaptive turbulent particle swarm optimization. In: Proceedings of fifth international conference on hybrid intelligent systems (HIS'05), Rio de Janeiro, Brazil, 6–9 November
55. Zahara E, Kao YT (2009) Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems. Expert Syst Appl 36:3880–3886
56. Qian X, Cao M, Su Z, Chen J (2012) A hybrid particle swarm optimization (PSO)-simplex algorithm for damage identification of delaminated beams. Math Probl Eng, Article ID 607418, p 11
57. Kaveh A, Talatahari S (2007) A discrete particle swarm ant colony optimization for design of steel frames. Asian J Civil Eng 9(6):563–575
58. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: Proceedings of the conference on systems, man and cybernetics, Piscataway, New Jersey, pp 4104–4109
59. Chen WN, Zhang J, Chung HSH, Zhong WL, Wu WG, Shi Y (2010) A novel set-based particle swarm optimization method for discrete optimization problems. IEEE Trans Evol Comput 14 (2):278–300
60. Gomes MH (2011) Truss optimization with dynamic constraints using a particle swarm algorithm. Expert Syst Appl 38:957–968
61. Grandhi RV, Venkayya VB (1988) Structural optimization with frequency constraints. AIAA J 26:858–866
62. Sedaghati R, Suleman A, Tabarrok B (2002) Structural optimization with frequency constraints using finite element force method. AIAA J 40:382–388
63. Wang D, Zha WH, Jiang JS (2004) Truss optimization on shape and sizing with frequency constraints. AIAA J 42:1452–1456
64. Lingyun W, Mei Z, Guangming W, Guang M (2005) Truss optimization on shape and sizing with frequency constraints based on genetic algorithm. J Comput Mech 25:361–368
65. Kaveh A, Zolghadr A (2011) Shape and size optimization of truss structures with frequency constraints using enhanced charged system search algorithm. Asian J Civil Eng 12:487–509
66. Kaveh A, Zolghadr A (2012) Truss optimization with natural frequency constraints using a hybridized CSS-BBBC algorithm with trap recognition capability. Comput Struct 102–103:14–27
67. Lin JH, Chen WY, Yu YS (1982) Structural optimization on geometrical configuration and element sizing with static and dynamic constraints. Comput Struct 15:507–515