We are IntechOpen,
the world's leading publisher of
Open Access books
Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS

BOOK
CITATION
INDEX

INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Particle Swarm Optimization in Structural Design

Ruben E. Perez[1] and Kamran Behdinan[2]
[1]University of Toronto, Institute for Aerospace Studies,
[2]Ryerson University, Department of Aerospace Engineering
Canada

1. Introduction

Optimization techniques play an important role as a useful decision making tool in the design of structures. By deriving the maximum benefits from the available resources, it enables the construction of lighter, more efficient structures while maintaining adequate levels of safety and reliability. A large number of optimization techniques have been suggested over the past decades to solve the inherently complex problem posed in structural design. Their scope varies widely depending on the type of structural problem to be tackled. Gradient-based methods, for example, are highly effectively in finding local optima when the design space is convex and continuous and when the design problem involves large number of design variables and constraints. If the problem constraints and objective function are convex in nature, then it is possible to conclude that the local optimum will be a global optimum. In most structural problems, however, it is practically impossible to check the convexity of the design space, therefore assuring an obtained optimum is the best possible among multiple feasible solutions. Global non-gradient-based methods are able to traverse along highly non-linear, non-convex design spaces and find the best global solutions. In this category many unconstrained optimization algorithms have been developed by mimicking natural phenomena such as Simulated Annealing (Kirkpatrick et al., 1983), Genetic Algorithms (Goldberg, 1989), and Bacterial Foraging (Passino, 2002) among others. Recently, a new family of more efficient global optimization algorithms have been developed which are better posed to handle constraints. They are based on the simulation of social interactions among members of a specific species looking for food sources. From this family of optimizers, the two most promising algorithms, which are the subject of this book, are Ant Colony Optimization (Dorigo, 1986), and Particle Swarm Optimization or PSO. In this chapter, we present the analysis, implementation, and improvement strategies of a particle swarm optimization suitable for constraint optimization tasks. We illustrate the functionality and effectiveness of this algorithm, and explore the effect of the different PSO setting parameters in the scope of classical structural optimization problems.

1.1 The Structural Design Problem

Before we describe the implementation of the particle swarm approach, it is necessary to define the general structural design problem to understand the different modification and

improvements made later to the basic algorithm. Mathematically, a structural design problem can be defined as:

$$min \quad f(x,p) \quad s.t. \quad x \in D$$
$$where \quad D = \left\{ x \mid x \in [x_l, x_u] \subset \Re^n, g_j(x,p) \leq 0 \, \forall j \in [1,m] \right\} \tag{1}$$

where a specific structural attribute (e.g. weight) is defined as an objective or merit function f which is maximized or minimized using proper choice of the design parameters. The design parameters specify the geometry and topology of the structure and physical properties of its members. Some of these are independent design variables (x) which are varied to optimize the problem; while others can be fixed value parameters (p). From the design parameters, a set of derived attributes are obtained some of which can be defined as behaviour constraints (g) e.g., stresses, deflections, natural frequencies and buckling loads etc., These behaviour parameters are functionally related through laws of structural mechanics to the design variables. The role of an optimization algorithm in structural design will be then to find the best combination of design variables that lead to the best objective function performance, while assuring all constraints are met.

2. The Particle Swarm Algorithm

The PSO algorithm was first proposed in 1995 by Kennedy and Eberhart. It is based on the premise that social sharing of information among members of a species offers an evolutionary advantage (Kennedy & Eberhart, 1995). Recently, the PSO has been proven useful on diverse engineering design applications such as logic circuit design (e.g. Coello & Luna, 2003), control design (e.g. Zheng et al., 2003) and power systems design (e.g. Abido, 2002) among others. A number of advantages with respect to other global algorithms make PSO an ideal candidate for engineering optimization tasks. The algorithm is robust and well suited to handle non-linear, non-convex design spaces with discontinuities. It is also more efficient, requiring a smaller number of function evaluations, while leading to better or the same quality of results (Hu et al., 2003; and Hassan et al., 2005). Furthermore, as we will see below, its easiness of implementation makes it more attractive as it does not require specific domain knowledge information, internal transformation of variables or other manipulations to handle constraints.

2.1 Mathematical Formulation

The particle swarm process is stochastic in nature; it makes use of a velocity vector to update the current position of each particle in the swarm. The velocity vector is updated based on the "memory" gained by each particle, conceptually resembling an autobiographical memory, as well as the knowledge gained by the swarm as a whole (Eberhart & Kennedy, 1995). Thus, the position of each particle in the swarm is updated based on the social behaviour of the swarm which adapts to its environment by returning to promising regions of the space previously discovered and searching for better positions over time. Numerically, the position x of a particle i at iteration $k+1$ is updated as:

$$x_{k+1}^i = x_k^i + v_{k+1}^i \Delta t \tag{2}$$

where v_{k+1}^i is the corresponding updated velocity vector, and Δt is the time step value typically considered as unity (Shi & Eberhart, 1998a). The velocity vector of each particle is calculated as:

$$v_{k+1}^i = wv_k^i + c_1 r_1 \frac{\left(p_k^i - x_k^i\right)}{\Delta t} + c_2 r_2 \frac{\left(p_k^g - x_k^i\right)}{\Delta t} \tag{3}$$

where v_k^i is the velocity vector at iteration k, p_k^i & p_k^g are respectively the best ever position of particle i and the global best position of the entire swarm up to current iteration k, and r represents a random number in the interval [0,1]. The remaining terms are configuration parameters that play an important role in the PSO convergence behaviour. The terms c_1 and c_2 represent "trust" settings which respectively indicate the degree of confidence in the best solution found by each individual particle (c_1 - cognitive parameter) and by the swarm as a whole (c_2 - social parameter). The final term w, is the inertia weight which is employed to control the exploration abilities of the swarm as it scales the current velocity value affecting the updated velocity vector. Large inertia weights will force larger velocity updates allowing the algorithm to explore the design space globally. Similarly, small inertia values will force the velocity updates to concentrate in the nearby regions of the design space.

Figure 1 illustrates the particle position and velocity update as described above in a two-dimensional vector space. Note how the updated particle position will be affected not only by its relationship with respect to the best swarm position but also by the magnitude of the configuration parameters.
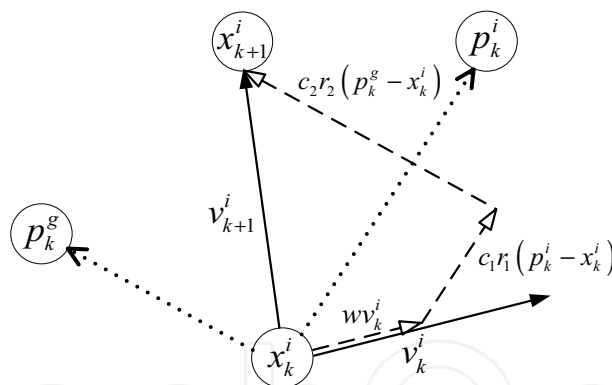

Figure 1. PSO Position and Velocity Update

2.2 Computational Algorithm

As with all numerical based optimization approaches the PSO process is iterative in nature, its basic algorithm is constructed as follows:

1. Initialize a set of particles positions x_0^i and velocities v_0^i randomly distributed throughout the design space bounded by specified limits.

2. Evaluate the objective function values $f\left(x_k^i\right)$ using the design space positions x_k^i. A total of n objective function evaluations will be performed at each iteration, where n is the total number of particles in the swarm.

3. Update the optimum particle position p_k^i at current iteration k and global optimum particle position p_k^g.
4. Update the position of each particle using its previous position and updated velocity vector as specified in Eq. (1) and Eq. (2).
5. Repeat steps 2-4 until a stopping criterion is met. For the basic implementation the typical stopping criteria is defined based on a number of iterations reached.

The iterative scheme behaviour for a two-dimensional variable space can be seen in Figure 2, where each particle position and velocity vector is plotted at two consecutive iterations. Each particle movement in the design space is affected based on its previous iteration velocity (which maintains the particle "momentum" biased towards a specific direction) and on a combined stochastic measure of the previous best and global positions with the cognitive and social parameters. The cognitive parameter will bias each particle position towards its best found solution space, while the social parameter will bias the particle positions towards the best global solution found by the entire swarm. For example, at the kth iteration the movement of the tenth particle in the figure is biased towards the left of the design space. However, a change in direction can be observed in the next iteration which is forced by the influence of the best design space location found by the whole swarm and represented in the figure as a black square. Similar behaviour can be observed in the other particles of the swarm.
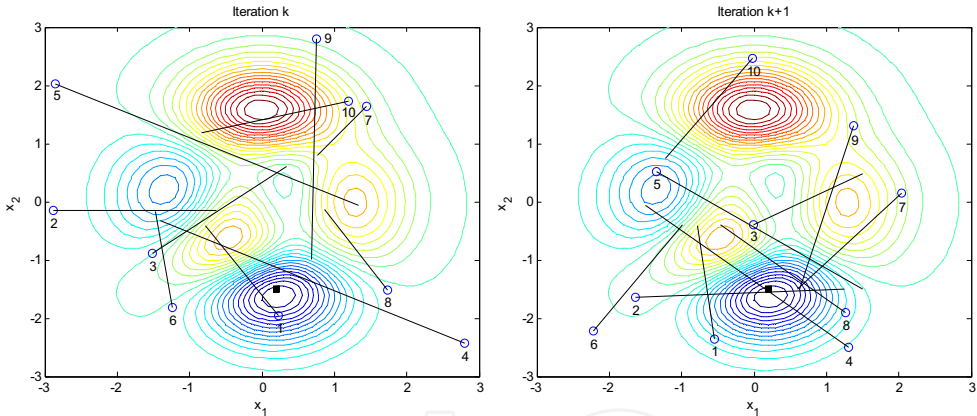

Figure 2. PSO Position and Velocity Update

An important observation is that the efficiency of the PSO is influenced to some extent by the swarm initial distribution over the design space. Areas not initially covered will only be explored if the momentum of a particle carries the particle into such areas. Such a case only occurs when a particle finds a new individual best position or if a new global best is discovered by the swarm. Proper setting of the PSO configuration parameters will ensure a good balance between computational effort and global exploration, so unexplored areas of the design space are covered. However, a good particle position initialization is desired. Different approaches have been used to initialize the particle positions with varying degrees of success. From an engineering design point of view, the best alternative will be to distribute particles uniformly covering the entire search space. A simpler alternative, which

has been proven successfully in practice, is to randomly distribute the initial position and velocity vectors of each particle throughout the design space. This can be accomplished using the following equations:

$$x_0^i = x_{min} + r\left(x_{max} - x_{min}\right) \tag{4}$$

$$v_0^i = \frac{x_{min} + r\left(x_{max} - x_{min}\right)}{\Delta t} \tag{5}$$

where x_{min} and x_{max} represent the lower and upper design variables bounds respectively, and r represents a random number in the interval [0,1]. Note that both magnitudes of the position and velocity values will be bounded, as large initial values will lead to large initial momentum and positional updates. This large momentum causes the swarm to diverge from a common global solution increasing the total computational time.

2.2 Algorithm Analysis

A useful insight of the PSO algorithm behaviour can be obtained if we replace the velocity update equation (Eq. (3)) into the position update equation (Eq. (2)) to get the following expression:

$$x_{k+1}^i = x_k^i + \left(wV_k^i + c_1 r_1 \frac{\left(p_k^i - x_k^i\right)}{\Delta t} + c_2 r_2 \frac{\left(p_k^g - x_k^i\right)}{\Delta t} \right) \Delta t \tag{6}$$

Factorizing the cognitive and social terms from the above equation we obtain the following general equation:

$$x_{k+1}^i = x_k^i + wV_k^i \Delta t + \left(c_1 r_1 + c_2 r_2\right) \left(\frac{c_1 r_1 p_k^i + c_2 r_2 p_k^g}{c_1 r_1 + c_2 r_2} - x_k^i \right) \tag{7}$$

Note how the above equation has the same structure as the gradient line-search used in convex unconstrained optimization ($x_{k+1}^i = \hat{x}_k^i + a\bar{p}_k$) where:

$$\hat{x}_k^i = x_k^i + wV_k^i \Delta t$$
$$a = c_1 r_1 + c_2 r_2 \tag{8}$$
$$\bar{p}_k = \frac{c_1 r_1 p_k^i + c_2 r_2 p_k^g}{c_1 r_1 + c_2 r_2} - x_k^i$$

So the behaviour of each particle in the swarm can be viewed as a traditional line-search procedure dependent on a stochastic step size (α) and a stochastic search direction ($\bar{p}_k$). Both the stochastic step size and search direction depend on the selection of social and cognitive parameters. In addition, the stochastic search direction behaviour is also driven by the best design space locations found by each particle and by the swarm as a whole. Behaviour confirmed from the Fig. 2 observations. Knowing that $r_1, r_2 \in [0,1]$, then the step size will belong to the interval $[0, c_1 + c_2]$ with a mean value of $(c_1 + c_2)/2$. Similarly, the search direction will be bracketed in the interval $\left[-x_k^i, \left(c_1 p_k^i + c_2 p_k^g / c_1 + c_2\right) - x_k^i t \right]$.

Two questions immediately arise from the above analysis. The first question is what type of convergence behaviour the algorithm will have. The second one is which values of the social and cognitive parameters will guarantee such convergence. To answer both questions let us start by re-arranging the position terms in equation (6) to get the general form for the ith particle position at iteration $k+1$ as:

$$x_{k+1}^i = x_k^i \left(1 - c_1 r_1 - c_2 r_2\right) + w V_k^i \Delta t + c_1 r_1 p_k^i + c_2 r_2 p_k^g \tag{9}$$

A similar re-arrangment of the position term in equation (2) leads to:

$$V_{k+1}^i = -x_k^i \frac{\left(c_1 r_1 + c_2 r_2\right)}{\Delta t} + w V_k^i + c_1 r_1 \frac{p_k^i}{\Delta t} + c_2 r_2 \frac{p_k^g}{\Delta t} \tag{10}$$

Equations (8) and (9) can be combined and written in matrix form as:

$$\begin{bmatrix} x_{k+1}^i \\ V_{k+1}^i \end{bmatrix} = \begin{bmatrix} 1 - c_1 r_1 - c_2 r_2 & w\Delta t \\ -\dfrac{\left(c_1 r_1 + c_2 r_2\right)}{\Delta t} & w \end{bmatrix} \begin{bmatrix} x_k^i \\ V_k^i \end{bmatrix} + \begin{bmatrix} c_1 r_1 & c_2 r_2 \\ \dfrac{c_1 r_1}{\Delta t} & \dfrac{c_2 r_2}{\Delta t} \end{bmatrix} \begin{bmatrix} p_k^i \\ p_k^g \end{bmatrix} \tag{11}$$

which can be considered as a discrete-dynamic system representation for the PSO algorithm where $\left[x^i, V^i\right]^T$ is the state subject to an external input $\left[p^i, p^g\right]^T$, and the first and second matrices correspond to the dynamic and input matrices respectively.

If we assume for a given particle that the external input is constant (as is the case when no individual or communal better positions are found) then a convergent behaviour can be maintained, as there is no external excitation in the dynamic system. In such a case, as the iterations go to infinity the updated positions and velocities will become the same from the kth to the $k^{th}+1$ iteration reducing the system to:

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -\left(c_1 r_1 + c_2 r_2\right) & w\Delta t \\ -\dfrac{\left(c_1 r_1 + c_2 r_2\right)}{\Delta t} & w - 1 \end{bmatrix} \begin{bmatrix} x_k^i \\ V_k^i \end{bmatrix} + \begin{bmatrix} c_1 r_1 & c_2 r_2 \\ \dfrac{c_1 r_1}{\Delta t} & \dfrac{c_2 r_2}{\Delta t} \end{bmatrix} \begin{bmatrix} p_k^i \\ p_k^g \end{bmatrix} \tag{12}$$

which is true only when $V_k^i = 0$ and both x_k^i and p_k^i coincide with p_k^g. Therefore, we will have an equilibrium point for which all particles tend to converge as iteration progresses. Note that such a position is not necessarily a local or global minimizer. Such point, however, will improve towards the optimum if there is external excitation in the dynamic system driven by the discovery of better individual and global positions during the optimization process.

The system stability and dynamic behaviour can be obtained using the eigenvalues derived from the dynamic matrix formulation presented in equation (11). The dynamic matrix characteristic equation is derived as:

$$\lambda^2 - \left(w - c_1 r_1 - c_2 r_2 + 1\right)\lambda + w = 0 \tag{13}$$

where the eigenvalues are given as:

$$\lambda_{1,2} = \frac{1 + w - c_1 r_1 - c_2 r_2 \pm \sqrt{\left(1 + w - c_1 r_1 - c_2 r_2\right)^2 - 4w}}{2} \tag{14}$$

The necessary and sufficient condition for stability of a discrete-dynamic system is that all eigenvalues (λ) derived from the dynamic matrix lie inside a unit circle around the origin on the complex plane, so $|\lambda_{i=1,\dots,n}| < 1$. Thus, convergence for the PSO will be guaranteed if the following set of stability conditions is met:

$$c_1 r_1 + c_2 r_2 > 0$$
$$\frac{(c_1 r_1 + c_2 r_2)}{2} - w < 1 \tag{15}$$
$$w < 1$$

Knowing that $r_1, r_2 \in [0,1]$ the above set of conditions can be rearranged giving the following set of parameter selection heuristics which guarantee convergence for the PSO:

$$0 < (c_1 + c_2) < 4$$
$$\frac{(c_1 + c_2)}{2} - 1 < w < 1 \tag{16}$$

While these heuristics provide useful selection bounds, an analysis of the effect of the different parameter settings is essential to determine the sensitivity of such parameters in the overall optimization procedure. Figure 3 shows the convergence histories for the well-known 10-bar truss structural optimization problem (described in more detail on Section 4) under different social and cognitive parameter combinations which meet the above convergence limits. The results are representative of more than 20 trials for each tested case, where the algorithm was allowed to run for 1000 iterations, with a fixed inertia weight value of 0.875, and the same initial particles, velocity values, and random seed.
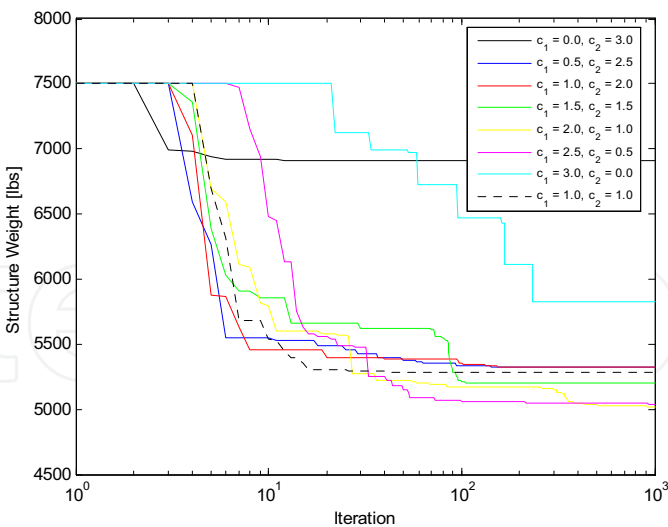

Figure 3. Cognitive (c_1) and Social (c_2) Parameters Variation Effect

From the figure we can clearly see that when only social values are used to update the particle velocities, as is the case when $c_1=0$ & $c_2=3$, the algorithm converges to a local optimum within the first ten iterations. As no individual (local) exploration is allowed to improve local solutions, all the particles in the swarm converge rapidly to the best initial optimum found from the swarm. We can also see that by increasing the emphasis on the cognitive parameter while reducing the social parameters, better solutions are found requiring less number of iterations for convergence. When we place slightly higher emphasis in the local exploration by each particle, as is the case with $c_1=2.0$ & $c_2=1.0$ and $c_1=2.5$ & $c_2=0.5$, the algorithm provides the best convergence speed to accuracy ratio. This result is due to the fact that individuals concentrate more in their own search regions thus avoiding overshooting the best design space regions. At the same time, some global information exchange is promoted, thus making the swarm point towards the best global solution. However, increasing local exploration at the expense of global agreement has its limits as shown in the case where only cognitive values are used to update the particle velocities ($c_1=3$ and $c_2=0$). In this case, each particle in the swarm will explore around its best-found solution requiring a very large number of iterations to agree into a common solution, which for this example is not the global optimum.

In a similar way to the above analysis, Figure 4 shows the effect of varying the inertia weight between its heuristic boundaries for a fixed set of "trust" settings parameters with $c_1=2.0$ and $c_2=1.0$ values. As before, the results are representative of more than 20 trials for each tested case where the algorithm was allowed to run for 1000 iterations with the same initial position, velocity and random seed. From the figure it is clear that reducing the inertia weight promotes faster convergence rates, as it controls the particle "momentum" bias towards a specific direction of the search space. Reducing the inertia weight beyond its allowable convergence limits comes at a cost as particles are forced to reduced their momentum stagnating at local optima as shown in the figure for the $w=0.5$ case.
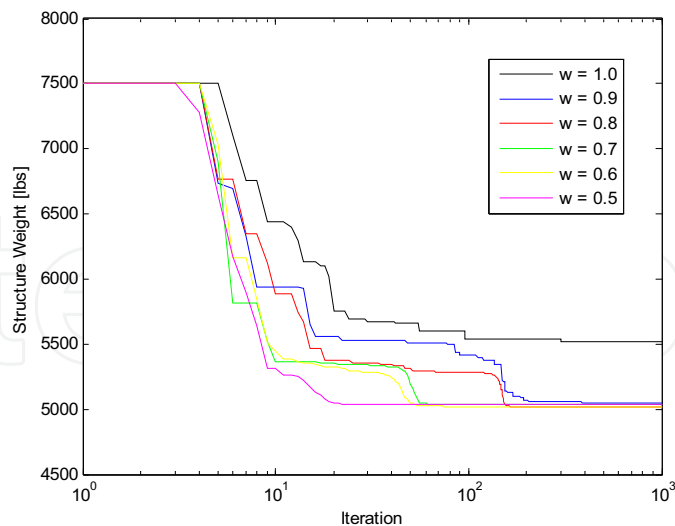

Figure 4. Inertia Weight Variation Effect

It is important to note at this stage that the optimal selection of the PSO parameters is in general problem-dependent. However, the obtained results for our example confirms the expectations as derived from the theoretical analysis (see i.e. van den Bergh & Engelbrecht, 2006; Trelea, 2003; and Clerc & Kennedy, 2002) and experiments (see i.e. Shi & Eberhart, 1998b; Shi & Eberhart, 1999; and Eberhart & Shi, 2000) regarding the sensitivity and behaviour of such tuning parameters. As long as the stability conditions presented in Eq. (16) are met, it is observed that maintaining an approximately equal or slightly higher weighting of the cognitive vs. the social parameter (in the interval of 1.5 to 2.5) will lead to the optimal convergent behaviour for the PSO.

3. Algorithm Improvements

Thus far, we have only dealt with the most basic PSO algorithm. Two important concerns when dealing with practical engineering problems have been left out up to now: how to improve the convergence rate behaviour as particles converge to a solution, and how to handle constraints. As we will see below, different modifications can be made to the original algorithm to address these concerns making it much stronger to deal with constrained optimization problems such as those traditionally present in structural design.

3.1 Updating the Inertia Weight

As shown before, the PSO global convergence is affected by the degree of local/global exploration provided by the "trust" settings parameters while the relative rate of convergence is provided by the inertia weight parameter. An interesting observation can be made from the inertia weight analysis presented in Figure 4. For a fixed inertia value there is a significant reduction in the algorithm convergence rate as iterations progresses. This is the consequence of excessive momentum in the particles, which results in detrimentally large steps sizes that overshoot the best design areas. By observing the figure, an intuitive strategy comes to mind: during the initial optimization stages, allow large weight updates so the design space is searched thoroughly. Once the most promising areas of the design space have been found (and the convergence rate starts to slow down) reduce the inertia weight, so the particles momentum decreases allowing them to concentrate in the best design areas. Formally, different methods have been proposed to accomplish the above strategy. Notably, two approaches have been used extensively (see Shi & Eberhart, 1998a; and Fourie & Groenwold, 2002). In the first one, a variation of inertia weight is proposed by linearly decreasing w at each iteration as:

$$w_{k+1} = w_{max} - \frac{w_{max} - w_{min}}{k_{max}} k \qquad (17)$$

where an initial inertia value w_{max} is linearly decreased during k_{max} iterations.
The second approach provides a dynamic decrease of the inertia weight value if the swarm makes no solution improvement after certain number of iterations. The updated is made from an initial weight value based on a fraction multiplier $k_w \in [0,1]$ as:

$$w_{k+1} = k_w w_k \qquad (18)$$

The effect of the described inertia update methods against a fixed inertia weight of 0.875 is shown in Figure 5 for the 10-bar truss example used before. For the linear decrease method, the inertia weight is varied in the interval [0.95, 0.55]. This interval meets the specified convergence conditions (Eq. (16)) with c_1=2.0 and c_2=1.0 values. For the dynamic decrease case a fraction multiplier of k_w = 0.975 is used if the improved solution does not change after five iterations, with an initial inertia weight specified as 0.95. As expected, an initial rapid convergence rate can be observed for the fixed inertia test, followed by a slow convergence towards the global optimum. The effect of dynamically updating the inertia weight is clear as both the linear and dynamic decrease methods present faster overall convergence rates. The dynamic update method provide the fastest convergence towards the solution, taking approximately 100 iterations as compared to 300 iterations taken by the linear decrease method, and the 600 iterations taken by the fixed inertia weight test. An intrinsic advantage is also provided by the dynamic decrease method as it depends solely on the value of past solutions adapting well to algorithmic termination and convergence check strategies.
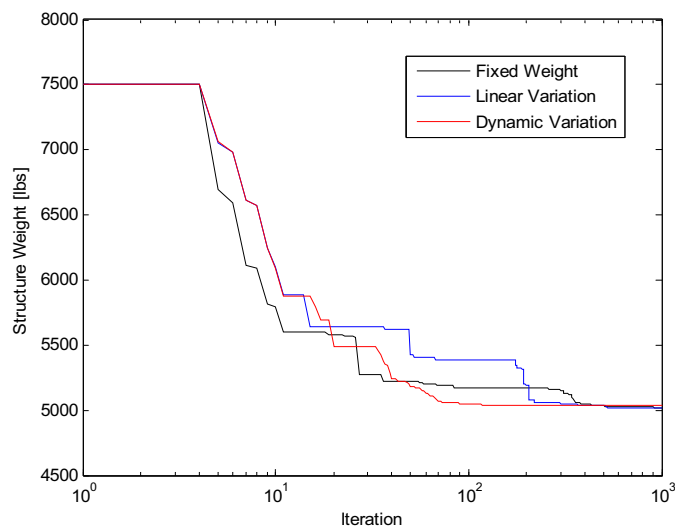

Figure 5. Inertia Weight Update Strategies Effect

3.2 Dealing with Constraints

Similar to other stochastic optimization methods, the PSO algorithm is formulated as an unconstrained optimizer. Different strategies have been proposed to deal with constraints, making the PSO a strong global engineering optimizer. One useful approach is to restrict the velocity vector of a constrained violated particle to a usable feasible direction as shown in Fig. 6. By doing so, the objective function is reduced while the particle is pointed back towards the feasible region of the design space (Venter & Sobieszczanski-Sobieski, 2004). A new position for the violated constraint particles can be defined using Eq. (2) with the velocity vector modified as:

$$v_{k+1}^i = c_1 r_1 \frac{\left(p_k^i - x_k^i\right)}{\Delta t} + c_2 r_2 \frac{\left(p_k^g - x_k^i\right)}{\Delta t} \tag{19}$$

where the modified velocity vector includes only the particle self information of the best point and the information of the current best point in the swarm. The new velocity vector is only influenced then by the particle best point found so far and by the current best point in the swarm. If both of these best points are feasible, the new velocity vector will point back to a feasible region of the design space. Otherwise, the new velocity vector will point to a region of the design space that resulted in smaller constraint violations. The result is to have the violated particle move back towards the feasible region of the design space, or at least closer to its boundary, in the next design iteration.
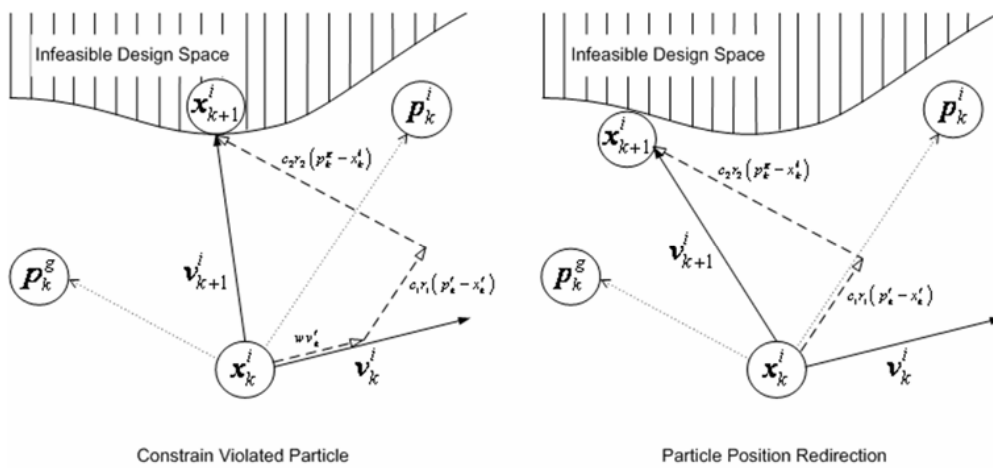

Figure 5. Violated Design Points Redirection

The velocity redirection approach however, does not guarantee that for the optimum solution all constraints will be met as it does not deal with the constraint directly. One classic way to accommodate constraints directly is by augmenting the objective function with penalties proportional to the degree of constraint infeasibility as:

$$f'(x_k) = \begin{cases} f(x_k) & if \quad x_k \quad is \quad feasible \\ f(x_k) + \sum_{j=1}^{m} k_j \hat{g}_j(x_k) & otherwise \end{cases} \tag{20}$$

where for m constraints k_j is a prescribed scaling penalty parameter and $\bar{g}_j(x_k)$ is a constraint value multiplier whose values are larger than zero if the constraint is violated:

$$\hat{g}_j(x_k) = \max\left(0, \left[g_j(x_k)\right]^2\right) \tag{21}$$

In a typical optimization procedure, the scaling parameter will be linearly increased at each iteration step so constraints are gradually enforced. The main concern with this method is that the quality of the solution will directly depend on the value of the specified scaling

parameters. A better alternative will be to accommodate constraints using a parameter-less scheme. Taking advantage of the available swarm information Eq. (22) and Eq. (23) show a useful adaptive penalty approach, where penalties are defined based on the average of the objective function and the level of violation of each constraint during each iteration step:

$$k_j = \left| \overline{f}(x_k) \right| \frac{\overline{g}_j(x_k)}{\sum_{l=1}^{m} \left[\overline{g}_l(x_k) \right]^2} \tag{22}$$

with

$$\overline{g}_j(x_k) = \frac{1}{n} \sum_{k=1}^{n} \max \left(0, \hat{g}_j(x_k) \right) \tag{23}$$

where $\overline{f}(x_k)$ is the average of the objective function values in the current swarm, and $\overline{g}_j(x_k)$ is the violation of the lth constraint averaged over the current population. The above formulation distributes the penalty coefficients in a way that those constraints which are more difficult to be satisfied will have a relatively higher penalty coefficient. Such distribution is achieved by making the jth coefficient proportional to the average of the violation of the jth constraint by the elements of the current population. An individual in the swarm whose j^{th} violation equals the average of the jth violation in the current population for all j, will have a penalty equal to the absolute value of the average objective function of the population. Similarly, the average of the objective function equals $\overline{f}(x_k) + \left| \overline{f}(x_k) \right|$.

While the penalty based method works well in many practical cases, the numerically exact constrained optimum feasible solution can only be obtained at the infinite limit of the penalty factor. Recently, a new approach which circumvents the need for infinite penalty factors has been proposed by Sedlaczek & Eberhard (2005). It directly uses the general Lagrange function defined for an ith particle as:

$$\Im_i\left(x_k^i, \lambda^i\right) = f\left(x_k^i\right) + \sum_{j=1}^{m} \lambda_j^i g_j\left(x_k^i\right) \tag{24}$$

where λ are Lagrange multipliers. This function can be used as an unconstrained pseudo objective function by realizing that the solution of a constrained optimization problem (Eq. (1)) with the correct set of multipliers is a stationary point for the function. The stationary is not necessarily a minimum of the Lagrange function. To preserve the stationary properties of the solution while assuring that it is a minimum, the Lagrange function is augmented using a quadratic function extension θ as (Gill et al. 1981):

$$\Im_i\left(x_k^i, \lambda^i, r_p^i\right) = f\left(x_k^i\right) + \sum_{j=1}^{m} \lambda_j^i \theta_j\left(x_k^i\right) + \sum_{j=1}^{m} r_{p,j} \theta_j^2\left(x_k^i\right) \tag{25}$$

with

$$\theta_j\left(x_k^i\right) = max\left[g_j\left(x_k^i\right), \frac{-\lambda_j}{2r_{p,i}} \right] \tag{26}$$

where the $-\lambda_j/2r_{p,i}$ term is arbitrarily chosen from gradient-based optimization problems. Note from Eq. (25) how each constraint violation is penalized separately using an r_p penalty factor. It can be shown that each constrain penalty factor will be of finite value in the solution of the augmented Lagrange function (Eq. (25)) hence in the solution of the original constrained problem (Eq. (1)). The multipliers and penalty factors values that lead to the optimum are unknown and problem dependent. Therefore, instead of the traditional single unconstrained optimization process, a sequence of unconstrained minimizations of Eq. (25) is required to obtain a solution. In such a sequence, the Lagrange multiplier is updated as:

$$\lambda_j^i\big|_{v+1} = \lambda_j^i\big|_v + 2\,r_{p,j}\big|_v\,\theta_j\left(x_k^i\right) \tag{27}$$

In a similar way, the penalty factor is updated in a way such that it penalizes infeasible movements as:

$$r_{p,j}\big|_{v+1} = \begin{cases} 2\,r_{p,j}\big|_v & if & g_j\left(x_v^i\right) > g_j\left(x_{v-1}^i\right) \wedge g_j\left(x_v^i\right) > \varepsilon_g \\ \dfrac{1}{2}r_{p,j}\big|_v & if & g_j\left(x_v^i\right) \le \varepsilon_g \\ r_{p,j}\big|_v & otherwise \end{cases} \tag{28}$$

where ε_g is a specified constrained violation tolerance. A lower bound limit of $r_{p,j} \ge (1/2)\sqrt{\left|\lambda_j^i\right|/\varepsilon_g}$ is also placed in the penalty factor so its magnitude is effective in creating a measurable change in Lagrange multipliers. Based on the above formulation the augmented Lagrange PSO algorithm can be then constructed as follows:

1. Initialize a set of particles positions x_0^i and velocities v_0^i randomly distributed throughout the design space bounded by specified limits. Also initialize the Lagrange multipliers and penalty factors, e.g. $\lambda_j^i\big|_0 = 0$, $r_{p,j}\big|_0 = r_0$, and evaluate the initial particles corresponding function values using Eq. (25).
2. Solve the unconstrained optimization problem described in Eq. (25) using the PSO algorithm shown in section 2.2 for k_{max} iterations.
3. Update the Lagrange multipliers and penalty factors according to Eq. (27) and Eq. (28).
4. Repeat steps 2-4 until a stopping criterion is met.

4. PSO Application to Structural Design

Particle swarms have not been used in the field of structural optimization until very recently, where they have show promising results in the areas of structural shape optimization (Fourie & Groenwold, 2002; Venter & Sobieszczanski-Sobieski, 2004) as well as topology optimization (Fourie & Groenwold, 2001). In this section, we show the application of the PSO algorithm to three classic non-convex truss structural optimization examples to demonstrate its effectiveness and to illustrate the effect of the different constraint handling methods.

4.1 Example 1 – The 10-Bar Truss

Our first example considers a well-known problem corresponding to a 10-bar truss non-convex optimization shown on Fig. 6 with nodal coordinates and loading as shown in Table 1 and 2 (Sunar & Belegundu, 1991). In this problem the cross-sectional area for each of the 10 members in the structure are being optimized towards the minimization of total weight. The cross-sectional area varies between 0.1 to 35.0 in². Constraints are specified in terms of stress and displacement of the truss members. The allowable stress for each member is 25,000 psi for both tension and compression, and the allowable displacement on the nodes is ±2 in, in the x and y directions. The density of the material is 0.1 lb/in³, Young's modulus is $E = 10^4$ ksi and vertical downward loads of 100 kips are applied at nodes 2 and 4. In total, the problem has a variable dimensionality of 10 and constraint dimensionality of 32 (10 tension constraints, 10 compression constraints, and 12 displacement constraints).
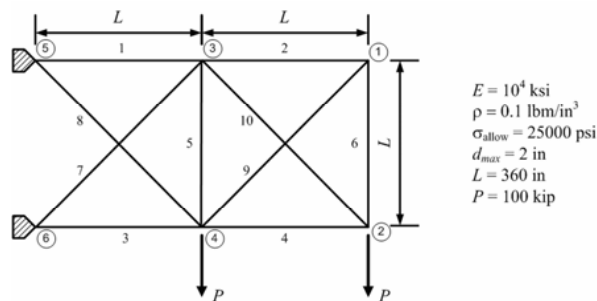

Figure 6. 10-Bar Space Truss Example

Node	x (in)	y (in)
1	720	360
2	720	0
3	360	360
4	360	0
5	0	360
6	0	0

Table 1. 10-Bar Truss Members Node Coordinates

Node	Fx	Fy
4	0	-100
6	0	-100

Table 2. 10-Bar Truss Nodal Loads

Three different PSO approaches where tested corresponding to different constraint handling methodologies. The first approach (PSO1) uses the traditional fixed penalty constraint while the second one (PSO2) uses an adaptive penalty constraint. The third approach (PSO3) makes use of the augmented Lagrange multiplier formulation to handle the constraints. Based on the derived selection heuristics and parameter settings analysis, a dynamic inertia weight variation method is used for all approaches with an initial weight of 0.95, and a fraction multiplier of $k_w = 0.975$ which updates the inertia value if the improved solution does not change after five iterations. Similarly, the "trust" setting parameters where specified

as $c_1=2.0$ and $c_2=1.0$ for the PSO1 and PSO2 approaches to promote the best global/local exploratory behaviour. For the PSO3 approach the setting parameters where reduced in value to $c_1=1.0$ and $c_2=0.5$ to avoid premature convergence when tracking the changing extrema of the augment multiplier objective function.

Table 3 shows the best and worst results of 20 independent runs for the different PSO approaches. Other published results found for the same problem using different optimization approaches including gradient based algorithms both unconstrained (Schimit & Miura, 1976), and constrained (Gellatly & Berke, 1971; Dobbs & Nelson, 1976; Rizzi, 1976; Haug & Arora, 1979; Haftka & Gurdal, 1992; Memari & Fuladgar, 1994), structural approximation algorithms (Schimit & Farshi, 1974), convex programming (Adeli & Kamal, 1991, Schmit & Fleury, 1980), non-linear goal programming (El-Sayed & Jang, 1994), and genetic algorithms (Ghasemi et al, 1997; Galante, 1992) are also shown in Tables 3 and 4.

Truss Area	PSO1 Best	PSO1 Worst	PSO2 Best	PSO2 Worst	PSO3 Best	PSO3 Worst	Gellatly & Berke, 1971	Schimit & Miura, 1976	Ghasemi, 1997	Schimit & Farshi, 1974	Dobbs & Nelson, 1976
01	33.50	33.50	33.50	33.50	33.50	30.41	31.35	30.57	25.73	33.43	30.50
02	0.100	0.100	0.100	0.100	0.100	0.380	0.100	0.369	0.109	0.100	0.100
03	22.76	28.56	22.77	33.50	22.77	25.02	20.03	23.97	24.85	24.26	23.29
04	14.42	21.93	14.42	13.30	14.42	14.56	15.60	14.73	16.35	14.26	15.43
05	0.100	0.100	0.100	0.100	0.100	0.110	0.140	0.100	0.106	0.100	0.100
06	0.100	0.100	0.100	0.100	0.100	0.100	0.240	0.364	0.109	0.100	0.210
07	7.534	7.443	7.534	6.826	7.534	7.676	8.350	8.547	8.700	8.388	7.649
08	20.46	19.58	20.47	18.94	20.47	20.83	22.21	21.11	21.41	20.74	20.98
09	20.40	19.44	20.39	18.81	20.39	21.21	22.06	20.77	22.30	19.69	21.82
10	0.100	0.100	0.100	0.100	0.100	0.100	0.100	0.320	0.122	0.100	0.100
Weight	5024.1	5405.3	5024.2	5176.2	5024.2	5076.7	5112.0	5107.3	5095.7	5089.0	5080.0

Table 3. 10-Bar Truss Optimization Results

Truss Area	Rizzi, 1976	Haug & Arora, 1979	Haftka & Gurdal, 1992	Adeli & Kamal, 1991	El-Sayed & Jang, 1994	Galante, 1992	Memari & Fuladgar, 1994
01	30.73	30.03	30.52	31.28	32.97	30.44	30.56
02	0.100	0.100	0.100	0.10	0.100	0.100	0.100
03	23.934	23.274	23.200	24.65	22.799	21.790	27.946
04	14.733	15.286	15.220	15.39	14.146	14.260	13.619
05	0.100	0.100	0.100	0.10	0.100	0.100	0.100
06	0.100	0.557	0.551	0.10	0.739	0.451	0.100
07	8.542	7.468	7.457	7.90	6.381	7.628	7.907
08	20.954	21.198	21.040	21.53	20.912	21.630	19.345
09	21.836	21.618	21.530	19.07	20.978	21.360	19.273
10	0.100	0.100	0.100	0.10	0.100	0.100	0.100
Weight	5061.6	5060.9	5060.8	5052.0	5013.2	4987.0	4981.1

Table 4. 10-Bar Truss Optimization Results (Continuation)

We can see that all three PSO implementations provide good results as compared with other methods for this problem. However, the optimal solution found by the fixed penalty approach has a slight violation of the node 3 and node 6 constraints. This behaviour is expected from a fixed penalty as the same infeasibility constraint pressure is applied at each iteration; it also indicates that either we should increase the scaling penalty parameter or dynamically increase it, so infeasibility is penalized further as the algorithm gets closer to the solution. The benefit of a dynamic varying penalty is demonstrated by the adaptive penalty PSO which meets all constraints and has only two active constraints for the displacements at nodes 3 and 6. The augmented Lagrange multiplier approach also converges to the same feasible point as the dynamic penalty result. Furthermore, it does it in fewer number of iterations as compared to the other two approaches since convergence is checked directly using the Lagrange multiplier and penalty factor values. Note as well how

the fixed penalty approach has a larger optimal solution deviation as compared to the dynamic penalty and Lagrange multiplier approaches.

4.2 Example 2 – The 25-Bar Truss

The second example considers the weight minimization of a 25-bar transmission tower as shown on Fig 7 with nodal coordinates shown on Table 5 (Schmit & Fleury, 1980). The design variables are the cross-sectional area for the truss members, which are linked in eight member groups as shown in Table 6. Loading of the structure is presented on Table 7. Constraints are imposed on the minimum cross-sectional area of each truss (0.01 in²), allowable displacement at each node (±0.35 in), and allowable stresses for the members in the interval [-40, 40] ksi. In total, this problem has a variable dimensionality of eight and a constraint dimensionality of 84.
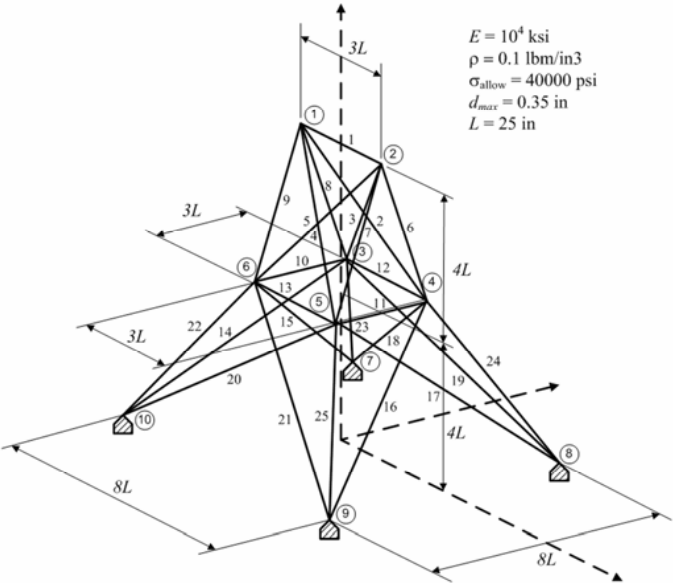

Figure 7. 25-Bar Space Truss Example

Node	x (in)	y (in)	z (in)
1	-37.5	0	200.0
2	37.5	0	200.0
3	-37.5	37.5	100.0
4	37.5	37.5	100.0
5	37.5	-37.5	100.0
6	-37.5	-37.5	100.0
7	-100.0	100.0	0.0
8	100.0	100.0	0.0
9	100.0	-100.0	0.0
10	-100.0	-100.0	0.0

Table 5. 25-Bar Truss Members Node Coordinates

Group	Truss Members
A1	1
A2	2-5
A3	6-9
A4	10,11
A5	12,13
A6	14-17
A7	18-21
A8	22-25

Table 6. 25-Bar Truss Members Area Grouping

Node	Fx	Fy	Fz
1	1000	-10000	-10000
2	0	-10000	-10000
3	500	0	0
6	600	0	0

Table 7. 25-Bar Truss Nodal Loads

As before, three different PSO approaches that correspond to different constraint handling methods were tested. The best and worst results of 20 independent runs for each tested method are presented on Table 8 as well as results from other research efforts obtained from local (gradient-based) and global optimizers. Clearly, all PSO approaches yield excellent solutions for both its best and worst results where all the constraints are met for all the PSO methods. The optimal solutions obtained have the same active constraints as reported in other references as follows: the displacements at nodes 3 and 6 in the Y direction for both load cases and the compressive stresses in members 19 and 20 for the second load case. As before, a larger solution deviation in the fixed penalty results is observed as compared to the other two PSO approaches. In addition, results from the augmented Lagrange method are obtained in less number of iterations as compared to the penalty-based approaches.

Area Group	PSO1 Best	PSO1 Worst	PSO2 Best	PSO2 Worst	PSO3 Best	PSO3 Worst	Zhou & Rosvany, 1993	Haftka & Gurdal, 1992	Erbatur, et al., 2000	Zhu, 1986	Wu, 1995
A1	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.010	0.010	0.1	0.1	0.1
A2	0.8977	0.1000	1.0227	0.9895	0.4565	1.0289	1.987	1.987	1.2	1.9	0.5
A3	3.4000	3.3533	3.4000	3.4000	3.4000	3.4000	2.994	2.991	3.2	2.6	3.4
A4	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.010	0.010	0.1	0.1	0.1
A5	0.1000	0.1000	0.1000	3.4000	1.9369	0.1000	0.010	0.012	1.1	0.1	1.5
A6	0.9930	0.7033	0.6399	0.6999	0.9647	0.8659	0.684	0.683	0.9	0.8	0.9
A7	2.2984	2.3233	2.0424	1.9136	0.4423	2.2278	1.677	1.679	0.4	2.1	0.6
A8	3.4000	3.4000	3.4000	3.4000	3.4000	3.4000	2.662	2.664	3.4	2.6	3.4
Weight	489.54	573.57	485.33	534.84	483.84	489.424	545.16	545.22	493.80	562.93	486.29

Table 8. 25-Bar Truss Optimization Results

4.3 Example 3 – The 72-Bar Truss

The final example deals with the optimization of a four-story 72-bar space truss as shown on Fig. 8. The structure is subject to two loading cases as presented on Table 9. The optimization objective is the minimization of structural weight where the design variables are specified as the cross-sectional area for the truss members. Truss members are linked in 16 member groups as shown in Table 10. Constraints are imposed on the maximum

allowable displacement of 0.25 in at the nodes 1 to 16 along the x and y directions, and a maximum allowable stress in each bar restricted to the range [-25,25] ksi. In total, this problem has a variable dimensionality of 16 and a constraint dimensionality of 264.

Load Case	Node	Fx	Fy	Fz
1	1	5	5	-5
2	1	0	0	-5
2	0	0	-5	0
3	0	0	-5	0
4	0	0	-5	0

Table 9. 72-Bar Truss Nodal Loads
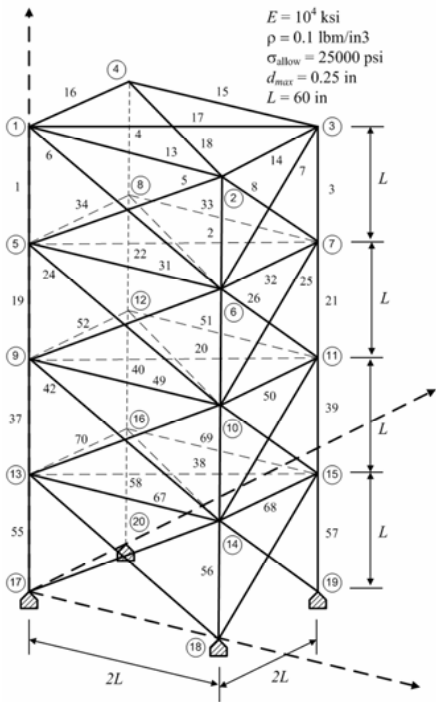
Figure 8. 72-Bar Space Truss Example

Results from the three PSO approaches as well as other references are shown in Table 11. As before, comparisons results include results from traditional optimization (Venkayya, 1971; Gellatly & Berke, 1971; Zhou & Rosvany, 1993), approximation concepts (Schimit & Farshi, 1974), and soft-computing approaches (Erbatur, et al., 2000). Similar to the previous examples, all the tested PSO approaches provide better solutions as those reported in the literature, with the augmented Lagrange method providing the best solution with the lowest number of iterations. The obtained optimal PSO solutions meet all constraints requirements and have the following active constraints: the displacements at node 1 in both the X and Y directions for load case one, and the compressive stresses in members 1-4 for load case two. The above active constraints agree with those reported by the different references.

Area Members Group	Truss Members
A1	1, 2, 3, 4
A2	5, 6, 7, 8, 9, 10, 11, 12
A3	13, 14, 15, 16
A4	17, 18
A5	19, 20, 21, 22
A6	23, 24, 25, 26, 27, 28, 29, 30
A7	31, 32, 33, 34
A8	35, 36
A9	37, 38, 39, 40
A10	41, 42, 43, 44, 45, 46, 47, 48
A11	49, 50, 51, 52
A12	53, 54
A13	55, 56, 57, 58
A14	59, 60, 61, 62, 63, 64, 65, 66
A15	67, 68, 69, 70
A16	71, 72

Table 10. 72-Bar Truss Members Area Grouping

Area Group	PSO1 Best	PSO1 Worst	PSO2 Best	PSO2 Worst	PSO3 Best	PSO3 Worst	Zhou & Rosvany, 1993	Venkayya, 1971	Erbatur, et al., 2000	Schimit & Farshi, 1974	Gellatly & Berke, 1971
A01	0.1561	0.1512	0.1615	0.1606	0.1564	0.1568	0.1571	0.161	0.155	0.1585	0.1492
A02	0.5708	0.5368	0.5092	0.5177	0.5553	0.5500	0.5356	0.557	0.535	0.5936	0.7733
A03	0.4572	0.4323	0.4967	0.3333	0.4172	0.3756	0.4096	0.377	0.480	0.3414	0.4534
A04	0.4903	0.5509	0.5619	0.5592	0.5164	0.5449	0.5693	0.506	0.520	0.6076	0.3417
A05	0.5133	2.5000	0.5142	0.4868	0.5194	0.5140	0.5067	0.611	0.460	0.2643	0.5521
A06	0.5323	0.5144	0.5464	0.5223	0.5217	0.4948	0.5200	0.532	0.530	0.5480	0.6084
A07	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.100	0.100	0.120	0.1000	0.1000
A08	0.1000	0.1000	0.1000	0.1095	0.1000	0.1000	0.100	0.100	0.165	0.1509	0.1000
A09	1.2942	1.2205	1.3079	1.3216	1.3278	1.2760	1.2801	1.246	1.155	1.1067	1.0235
A10	0.5426	0.5041	0.5193	0.5065	0.4998	0.4930	0.5148	0.524	0.585	0.5793	0.5421
A11	0.1000	0.1000	0.1000	0.1000	0.1000	0.1005	0.1000	0.100	0.100	0.1000	0.1000
A12	0.1000	0.1000	0.1000	0.1000	0.1000	0.1005	0.1000	0.100	0.100	0.1000	0.1000
A13	1.8293	1.7580	1.7427	2.4977	1.8992	2.2091	1.8973	1.818	1.755	2.0784	1.4636
A14	0.4675	0.4787	0.5185	0.4833	0.5108	0.5145	0.5158	0.524	0.505	0.5034	0.5207
A15	0.1003	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.100	0.105	0.1000	0.1000
A16	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.1000	0.100	0.155	0.1000	0.1000
Weight	381.03	417.45	381.91	384.62	379.88	381.17	379.66	381.20	385.76	388.63	395.97

Table 11. 72-Bar Truss Optimization Results

7. Summary and Conclusions

Particle Swarm Optimization is a population-based algorithm, which mimics the social behaviour of animals in a flock. It makes use of individual and group memory to update each particle position allowing global as well as local search optimization. Analytically the PSO behaves similarly to a traditional line-search where the step length and search direction are stochastic. Furthermore, it was shown that the PSO search strategy can be represented as a discrete-dynamic system which converges to an equilibrium point. From a stability analysis of such system, a parameter selection heuristic was developed which provides an initial guideline to the selection of the different PSO setting parameters. Experimentally, it was found that using the derived heuristics with a slightly larger cognitive pressure value

leads to faster and more accurate convergence. Improvements of the basic PSO algorithm were discussed. Different inertia update strategies were presented to improve the rate of convergence near optimum points. It was found that a dynamic update provide the best rate of convergence overall. In addition, different constraint handling methods were shown. Three non-convex structural optimization problems were tested using the PSO with a dynamic inertia update and different constraint handling approaches. Results from the tested examples illustrate the ability of the PSO algorithm (with all the different constraint handling strategies) to find optimal results, which are better, or at the same level of other structural optimization methods. From the different constraint handling methods, the augmented Lagrange multiplier approach provides the fastest and more accurate alternative. Nevertheless implementing such method requires additional algorithmic changes, and the best combination of setting parameters for such approach still need to be determined. The PSO simplicity of implementation, elegant mathematical features, along with the lower number of setting parameters makes it an ideal method when dealing with global non-convex optimization tasks for both structures and other areas of design.

8. References

Abido M. (2002). Optimal design of power system stabilizers using particle swarm optimization, *IEEE Trans Energy* Conversion, Vol. 17, No. 3, pp. 406–413.

Adeli H, and Kamal O. (1991).   Efficient optimization of plane trusses, *Adv Eng Software*, Vol. 13, No. 3, pp. 116–122.

Clerc M, and Kennedy J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space, *IEEE Trans Evolut. Comput.*, Vol. 6, No. 1, pp 58–73.

Coello C, and Luna E. (2003). Use of particle swarm optimization to design combinational logic Circuits, Tyrell A, Haddow P, Torresen J, editors. *5th International conference on evolvable systems: from biology to hardware, ICES 2003. Lecture notes in computer science*, vol. 2606. Springer, Trondheim, Norway, pp. 398–409.

Dobbs M, and Nelson R. (1976). Application of optimality criteria to automated structural design, *AIAA Journal*, Vol. 14, pp. 1436–1443.

Dorigo M, Maniezzo V, and Colorni A. (1996). The ant system: optimization by a colony of cooperating agents, *IEEE Trans Syst Man Cybernet B*, Vol. 26, No. 1, pp. 29–41.

Eberhart R, and Kennedy J. (1995). New optimizer using particle swarm theory, *Sixth Int. symposium on micro machine and human science*, Nagoya, Japan, pp. 39–43.

Eberhart R, and Shi Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization, *IEEE congress on evolutionary computation (CEC 2000)*, San Diego, CA, pp. 84–88.

El-Sayed M, and Jang T. (1994). Structural optimization using unconstrained non-linear goal programming algorithm, *Comput Struct*, Vol. 52, No. 4, pp. 723–727.

Erbatur F, Hasantebi O, Tntnncn I, Kilt H. (2000). Optimal design of planar and space structures with genetic algorithms, *Comput Struct*, Vol. 75, pp. 209–24.

Fourie P, and Groenwold A. (2001). The particle swarm optimization in topology optimization, *Fourth world congress of structural and multidisciplinary optimization*, Paper no. 154, Dalian, China.

Fourie P, and Groenwold A. (2002). The particle swarm optimization algorithm in size and shape optimization, *Struct Multidiscip Optimiz*, Vol. 23, No. 4, pp. 259–267.

Galante M. (1992). Structures Optimization by a simple genetic algorithm, *Numerical methods in engineering and applied sciences*, Barcelona, Spain, pp. 862–70.

Gellatly R, and Berke L. (1971). Optimal structural design, *Tech Rep AFFDLTR-70-165*, Air Force Flight Dynamics Laboratory (AFFDL).

Ghasemi M, Hinton E, and Wood R. (1997). Optimization of trusses using genetic algorithms for discrete and continuous variables, *Eng Comput*, Vol. 16, pp. 272–301.

Gill P.E., Murray W., and Wright M.H. (1981). *Practical Optimization*. Academic Press, London.

Goldberg D. (1989). Genetic algorithms in search, optimization, and machine learning. Addison- Wesley, New York, NY.

Haftka R, and Gurdal Z. (1992). *Elements of structural optimization*, 3rd ed. Kluwer Academic Publishers.

Hassan R, Cohanim B, de Weck O, and Venter G. (2005). A comparison of particle swarm optimization and the genetic algorithm, *1st AIAA multidisciplinary design optimization specialist conference,* Paper No. AIAA-2005-1897, Austin, TX.

Haug E, and Arora J. (1979). *Applied optimal design*, Wiley, New York, NY. Hu X, Eberhart R, and Shi Y. (2003). Engineering optimization with particle swarm, *IEEE Swarm intelligence symposium (SIS 2003),* Indianapolis, IN, pp. 53–57.

Kennedy J, and Eberhart R. (1995). Particle swarm optimization, *IEEE international conference on neural networks*, Vol. IV, Piscataway, NJ, pp. 1942–1948.

Kirkpatrick S, Gelatt C, and Vecchi M. (1983). Optimization by simulated annealing. *Science*, Vol. 220, No. 4598, pp. 671–680.

Memari A, and Fuladgar A. (1994). Minimum weight design of trusses by behsaz program, *2nd International conference on computational structures technology*, Athens, Greece.

Passino K. (2002). Biomimicry of bacterial foraging for distributed optimization and control, *IEEE Control Syst Mag*, Vol. 22, No. 3, pp. 52–67.

Rizzi P. (1976). Optimization of multiconstrained structures based on optimality criteria, 17th *Structures, structural dynamics, and materials conference*, King of Prussia, PA.

Schimit L, and Farshi B. (1974). Some approximation concepts in structural synthesis, *AIAA Journal*, Vol. 12, pp. 692–699.

Schimit L, and Miura H. (1976). A new structural analysis/synthesis capability: access 1. *AIAA Journal*, Vol. 14, pp. 661–671.

Schmit L, and Fleury C. (1980). Discrete-continuous variable structural synthesis using dual methods, *AIAA Journal*, Vol. 18, pp. 1515–1524.

Sedlaczek K. and Peter Eberhard (2005). Constrained Particle Swarm Optimization of Mechanical Systems, *6th World Congresses of Structural and Multidisciplinary Optimization*, Rio de Janeiro, 30 May - 03 June 2005, Brazil

Shi Y, and Eberhart R. (1998a). A modified particle swarm optimizer, *IEEE international conference on evolutionary computation*, IEEE Press, Piscataway, NJ, pp. 69–73.

Shi Y, and Eberhart R. (1998b). Parameter selection in particle swarm optimization, *7th annual conference on evolutionary programming*, New York, NY, pp. 591–600.

Shi Y, and Eberhart R. (1999). Empirical study of particle swarm optimization, *IEEE congress on evolutionary computation (CEC 1999)*, Piscataway, NJ, pp. 1945–50.

Sunar M, and Belegundu A. (1991). Trust region methods for structural optimization using exact second order sensitivity, *Int J. Numer Meth Eng*, Vol. 32, pp. 275–293.

Trelea I. (2003). The particle swarm optimization algorithm: convergence analysis and parameter selection, *Information Processing Letters*, Vol. 85, No. 6, pp. 317–325.

Van den Bergh F, and Engelbrecht A. (2006). A study of particle swarm optimization particle trajectories, *Information Science*, Vol. 176, pp. 937-971.

Venkayya V. (1971). Design of optimum structures, *Comput Struct*, Vol. 1, No. 12, pp. 265–309.

Venter G, and Sobieszczanski-Sobieski J. (2003). Particle swarm optimization, *AIAA Journal*, Vol. 41, No. 8, pp 1583–1589.

Venter G, and Sobieszczanski-Sobieski J. (2004). Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization, *Struct Multidiscip Optimiz*, Vol. 26, No. 1–2, pp. 121–131.

Wu S, Chow P. (1995). Steady-state genetic algorithms for discrete optimization of trusses, *Comput Struct*, Vol. 56, No. 6, pp. 979–991.

Zheng Y, Ma L, Zhang L, Qian J. (2003). Robust pid controller design using particle swarm optimizer, *IEEE international symposium on intelligence control*, pp. 974–979.

Zhou M, Rozvany G. (1993). Dcoc: an optimality criteria method for large systems. Part ii: algorithm, *Struct Multidiscip Optimiz*, Vol. 6, pp. 250–62.

Zhu D. (1986). An improved templemans algorithm for optimum design of trusses with discrete member sizes, *Eng Optimiz*, Vol. 9, pp. 303–312.
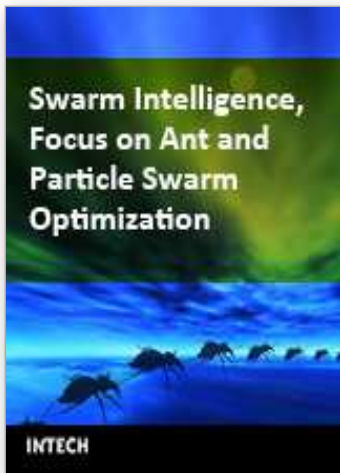
Swarm Intelligence, Focus on Ant and Particle Swarm Optimization

Edited by FelixT.S.Chan and Manoj KumarTiwari

In the era globalisation the emerging technologies are governing engineering industries to a multifaceted state. The escalating complexity has demanded researchers to find the possible ways of easing the solution of the problems. This has motivated the researchers to grasp ideas from the nature and implant it in the engineering sciences. This way of thinking led to emergence of many biologically inspired algorithms that have proven to be efficient in handling the computationally complex problems with competence such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), etc. Motivated by the capability of the biologically inspired algorithms the present book on "Swarm Intelligence: Focus on Ant and Particle Swarm Optimization" aims to present recent developments and applications concerning optimization with swarm intelligence techniques. The papers selected for this book comprise a cross-section of topics that reflect a variety of perspectives and disciplinary backgrounds. In addition to the introduction of new concepts of swarm intelligence, this book also presented some selected representative case studies covering power plant maintenance scheduling; geotechnical engineering; design and machining tolerances; layout problems; manufacturing process plan; job-shop scheduling; structural design; environmental dispatching problems; wireless communication; water distribution systems; multi-plant supply chain; fault diagnosis of airplane engines; and process scheduling. I believe these 27 chapters presented in this book adequately reflect these topics.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

INTECH
open science | open minds