Engineering Optimization
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/geno20

Enhancing particle swarm optimization algorithm using two new strategies for optimizing design of truss structures

Y. C. Lu [a] , J. C. Jan [b] , S. L. Hung [a] & G. H. Hung [a]
[a] Department of Civil Engineering , National Chiao Tung University , Hsinchu , Taiwan, ROC
[b] Department of Computer Science and Information Engineering , Chien Hsin University of Science and Technology , Jhongli , Taiwan, ROC
Published online: 03 Dec 2012.

To cite this article: Y. C. Lu , J. C. Jan , S. L. Hung & G. H. Hung (2013) Enhancing particle swarm optimization algorithm using two new strategies for optimizing design of truss structures, Engineering Optimization, 45:10, 1251-1271, DOI: 10.1080/0305215X.2012.729054

To link to this article: http://dx.doi.org/10.1080/0305215X.2012.729054

Engineering Optimization, 2013
Vol. 45, No. 10, 1251–1271, http://dx.doi.org/10.1080/0305215X.2012.729054

Enhancing particle swarm optimization algorithm using two new strategies for optimizing design of truss structures

Y.C. Lu[a], J.C. Jan[b], S.L. Hung[a]* and G.H. Hung[a]

[a]*Department of Civil Engineering, National Chiao Tung University, Hsinchu, Taiwan, ROC;*
[b]*Department of Computer Science and Information Engineering, Chien Hsin University of Science and Technology, Jhongli, Taiwan, ROC*

(*Received 27 March 2012; final version received 21 August 2012*)

This work develops an augmented particle swarm optimization (AugPSO) algorithm using two new strategies,: *boundary-shifting* and *particle-position-resetting*. The purpose of the algorithm is to optimize the design of truss structures. Inspired by a heuristic, the *boundary-shifting* approach forces particles to move to the boundary between feasible and infeasible regions in order to increase the convergence rate in searching. The purpose of the *particle-position-resetting* approach, motivated by mutation scheme in genetic algorithms (GAs), is to increase the diversity of particles and to prevent the solution of particles from falling into local minima. The performance of the AugPSO algorithm was tested on four benchmark truss design problems involving 10, 25, 72 and 120 bars. The convergence rates and final solutions achieved were compared among the simple PSO, the PSO with passive congregation (PSOPC) and the AugPSO algorithms. The numerical results indicate that the new AugPSO algorithm outperforms the simple PSO and PSOPC algorithms. The AugPSO achieved a new and superior optimal solution to the 120-bar truss design problem. Numerical analyses showed that the AugPSO algorithm is more robust than the PSO and PSOPC algorithms.

Keywords: particle swarm optimization (PSO); optimization design; truss structures; stochastic search method

1. Introduction

In the past two decades, stochastic search methods based on natural phenomena have been widely used to solve structural optimization problems. Such methods include the genetic algorithm (GA) (Wu and Chow 1995), the ant colony optimization (ACO) algorithm (Charles and Barron 2004, Kaveh and Shojaee 2007) and the particle swarm optimization (PSO) algorithm (Fourie and Groenwold 2002, Schutte and Groenwold 2003, Li *et al.* 2006, Liu *et al.* 2006 Fan and Chiu 2007, Fan and Chang 2009, 2010, Montazeri-Gh *et al.* 2012). Unlike conventional mathematical optimization approaches, these stochastic-based approaches can find a solution without gradient information and they therefore have a greater global search capacity. Despite their clear advantages, they also have some drawbacks. The convergence of binary GA is slow and the result may not be the optimal solution. The ACO has a longer search time than other methods and tends to terminate at a non-optimal solution. Whereas PSO has a high convergence rate, it easily falls into a

*Corresponding author. Email: slhung@mail.nctu.edu.tw

local optimum in non-convex problems. The PSO algorithm, which is a probabilistic and iterative approach, finds the optimal position in search space by simulating the behaviour of a flock of foraging birds. Meanwhile, the PSO algorithm has fewer parameters and is easier to implement than the GAs. For some problems, it also has a higher convergence rate than other evolutionary algorithms (Kennedy *et al.* 2001).

The particle swarm originated as a model of social flocking behaviour. Reynolds (1987) was fascinated by the aesthetics of bird flocking and used a particle system to capture this beautiful phenomenon as a computer animation. Heppner and Grenander (1990) sought the underlying rules that govern the synchronous flocking of numerous birds. The synchrony of flocking has been thought to be a function of the efforts of birds to maintain an optimal distance between themselves and adjacent birds. Kennedy and Eberhart (1995) developed the particle swarm optimization algorithm based on the synchrony of flocking behaviour and the assumption that individual members of a flock share social information and experience in their search for food. The PSO algorithm considers a population of particles as a group of potential solutions, and moves them around in the search space to find the optimal solution. The motions of the particles are determined from the best positions in the search space, which are updated whenever better positions are found by the particles from time step to time step.

The PSO algorithms have proven effective in many applications and fields, including the training of game agents, image and data clustering, power systems, applied mathematics, design optimization, controller design, bioinformatics, data mining, and others (Engelbrecht 2005). These algorithms can quickly obtain robust solutions to nonlinear, non-differentiable and multi-modal problems (Shi and Eberhart 1998). Exploration and exploitation are two search strategies for solving an optimization problem. The PSO balances exploration (global search) with exploitation (local search) by selecting an appropriate PSO algorithm and parameters (Binkley and Hagiwara 2008). Although the PSO algorithm can rapidly converge in the early searching stage, premature convergence may cause the particle searching procedure to fall into a local optimum.

Most modifications to the simple PSO have been made to improve its convergence rate and to increase the diversity of the swarm. Shi and Eberhart (1998) enhanced the effectiveness of the particle swarm optimizer by introducing the inertia weight concept to balance exploration and exploitation. The inertia weight can be a positive constant or a positive function of time. A properly chosen inertia weight ensures that the optimal solution is reached rapidly. Clerc (1999) introduced a constriction particle swarm optimization (CPSO) that included a constriction factor for increasing the convergence capacity of a local search. Eberhart and Shi (2000) compared CPSO and PSO with methods based on inertia weight and concluded that the best method is to apply the constriction factor while limiting the maximum velocity to a dynamic range in each dimension. He *et al.* (2004b) introduced the concept of passive congregation, which affects particle velocity according to the positions of other randomly selected particles. A particle swarm optimizer with passive congregation (PSOPC) can improve the search efficiency and the probability of finding the optimal solution. Shelokar *et al.* (2007) proposed a particle swarm ant colony optimization (PSACO) approach that combines PSO for global search and ACO for local search. The PSACO uses PSO for global optimization and uses an ant colony approach for updating particle positions so that the feasible solution space can be rapidly identified. A heuristic particle swarm optimizer (HPSO) introduced by Li *et al.* (2007), which combined PSOPC with harmony search, effectively increased the convergence rate in the early stage and reached the optimal design more rapidly than PSO or PSOPC. Kaveh and Talatahari (2009b) proposed a heuristic particle swarm ant colony optimization (HPSACO) based on PSOPC, ant colony optimization and a harmony search scheme. Their comparison results showed that their HPSACO has better efficiency and robustness than other PSO-based algorithms and has a higher convergence rate than PSO and PSOPC.

Mutation is a powerful strategy employed in GA to maintain the diversity of the population and to extend the exploration domain. Many mutation operators have been applied to PSO to increase particle diversity in a swarm. A Cauchy mutation operator introduced by Stacey *et al.* (2003) significantly improved performance in all dimensions of the Rastrigin and Rosenbrock functions and in dimensions 20 and 30 of the Ackley function. The Cauchy distribution resembles a normal distribution except for its higher probability in the tails. This increases the probability of generating large values. Andrews (2006) used various mutation operators to compare performance in solving optimization problems. His experimental results showed that including mutation operators in PSO improves optimization performance in both the local and global versions; however, the performance enhancement in each mutation operation depends on the difficulty of the optimization problem. Ling *et al.* (2008) applied a wavelet theory-based mutation operation to improve solution quality by enhancing the efficiency of the PSO in exploring the solution space. Wang *et al.* (2010) adopted a unified tabu and mutation framework to increase search diversity and avoid stagnation in local optima. Only the most frequently tabued particle is mutated in the mutation phase of the proposed framework.

Throughout the particle searching procedure, balancing exploration and exploitation is crucial for an effective optimization algorithm. The PSO has difficulty balancing global and local searches. The goal of this work is to develop an augmented particle swarm optimization (AugPSO) algorithm with an increased convergence rate in early search and increased diversity that does not fall into a local optimum. The two major strategies used in the AugPSO algorithm are heuristic-inspired *boundary-shifting* and mutation-like *particle-position-resetting*. The optimal solution to constrained optimization problems is usually in the critical boundaries between the feasible and infeasible regions (Singh *et al*. 2009). The boundary-shifting strategy, which is inspired by heuristics, moves the particles near the critical area in the early iterations and increases the early convergence rate. Motivated by mutation in GA, the particle-position-resetting strategy randomly resets the particle position farther away from the current centre of convergence to maintain the particle diversity and to avoid premature convergence to local minima. This study applies the AugPSO algorithm to the problem of optimizing truss design. The performance of the AugPSO algorithm is tested in four examples. The numerical convergence rate and final results are compared with those of simple PSO, PSOPC and other algorithms in the literature. A stochastic analysis is also performed to evaluate the performance of the algorithm.

2. Continuous design variables problem

This investigation presents a general truss structure design optimization problem under displacement and stress constraints. The general form is as follows:

$$\begin{aligned} &\min\ O(X) \\ &\text{subject to } h_j(X) \le e_j \quad \text{for } j = 1, 2, \ldots, J \\ &\qquad x_n^l \le x_n \le x_n^u \quad \text{for } n = 1, 2, \ldots, N \end{aligned}$$

where the objective function $O(X)$ is the total weight of the truss. The function $h_j(X)$ and e_j are the jth inequality constraint function and its predefined specified threshold, respectively. The decision variable X is composed of N design variables, $X = \{x_1, x_2, \ldots, x_N\}$, which are the cross-sectional areas of the bar members of a truss. These design variables define the design space; the objective function is a 'surface' of N dimensions, embedded in a space of $N + 1$ dimensions. The design variable x_n represents the cross-sectional area of the nth member of a truss. The lower and upper

limits on the design variable x_n are x_n^l and x_n^u, respectively. An available solution region (problem search space) for the optimization problem is defined in terms of these limits. Inequality constraints can then be applied to reduce the size of the feasible region.

3. Augmented particle swarm optimization algorithm

The AugPSO algorithm is based on simple PSO but applies two new mechanisms for enhancing its convergence rate and particle diversity. The boundary-shifting approach is employed to move particles close to the critical boundaries and to reduce the search time in the optimization process. The particle-position-resetting mechanism is utilized to maintain the diversity of solutions in the premature convergence process and to ensure that the final solution does not fall into a local optimum. The following sections describe the simple PSO, the new boundary-shifting strategy and the particle-position-resetting strategy.

3.1. *Particle swarm optimization algorithm*

The PSO developed by Kennedy and Eberhart (1995) is a population-based metaheuristic search method that uses swarm intelligence. The algorithm comprises a population of particles that are initiated with random potential solutions (positions). As these particles systematically move around the problem search space, each generates a new position according to an inertial velocity vector and two experiences, its own search experience (including the best position found in earlier searches) and the experience of the swarm (including the optimal solution currently captured by the population). The respective equations used for updating particle velocities and positions are:

$$V_i^{k+1} = wV_i^k + c_1 r_1 (P_i^k - X_i^k) + c_2 r_2 (P_G^k - X_i^k) \tag{1}$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \quad \text{for } i = 1, 2, \ldots, S \tag{2}$$

where k is the number of iterations; i is the number of particles in a given swarm with S particles; X_i^k and V_i^k represent the position and velocity, respectively, of the ith particle in the kth iteration; w is a given constant, denoting the inertia weight; c_1 and c_1 are positive constants for cognitive and social scaling parameters, respectively; r_1 and r_1 are two working variables generated using a uniform random function in the range [0,1]; P_i^k is the best position of the ith particle up to the kth iteration, and P_G^k is the best global position at present among all particles in the swarm. The objective function determines the particle solution quality. In Equation (1), the first term is particle velocity, which is determined by particle inertia; the second term represents particle cognition; the third term is the social activity of the swarm. The last two terms determine the change of position in each time interval iteration, and both are velocity terms.

The problem of optimizing truss design is formulated by using a swarm of particles to denote a group of S truss structures. Position X_i^k of a particle in the swarm represents the cross-sectional areas of truss members of the ith structure in the kth iteration; the best position (solution) P_i^k of ith particle (structure) can be obtained by applying the objective function to find the total weight of the truss and then comparing it with P_i^{k-1}. The best global solution P_G^k in the kth iteration is the best solution among all P_i^k for $i = 1, 2, \ldots, S$. Since the current optimal structure is equivalent to P_G^k, the new velocity V_i^{k+1} and position X_i^{k+1} can be obtained by the above equations. Compared to its previous position, the particle should then be closer to the optimal solution.

3.2. *Boundary-shifting strategy*

Relative minima are found more often than the optimal solution to a constrained optimization problem. The minima may be at a gradient of zero for the objective function or at a constraint boundary (Rao 2009). Assuming that particles can rapidly move to the boundaries, solution optimization is faster than that in a simple PSO. When a simple PSO algorithm is utilized, the particles move slowly to a better solution in a manner dependent on their objective function. This process is very time-consuming in the earlier stages of the search. In the truss optimization problem, the goal is minimizing the weight of the truss. When no constraints are imposed, the lightest structure has the smallest cross-sectional areas, which are equal to the lower limits, x_n^l, of design variables. However, constraints on stress and on displacement are usually applied when optimizing truss structure design. The optimal solution to this problem is possible either at the boundary of one constraint or at the intersection of the two constraints in some specific cases.

For any particle, the corresponding truss nodal displacement and member stress can be determined by structural analysis. In a truss structure analysis of size optimization problems, the cross-sectional area of the truss members is the only variable in the stiffness matrix because the topology and material properties of the truss are constant. Theoretically, an increase in all cross-sectional areas of the structure with the same ratio results in a reduction in nodal displacements and member stresses. Therefore, the cross-sectional area of the truss members can be adjusted to maximize the nodal displacements or the member stresses. In each iteration, all constraint functions for design variable X_i, *e.g.* e'_{ij}, must be solved and compared with the corresponding predefined constraint thresholds, *e.g.* e_j. The constraint functions may denote displacement or stress constraint dependent on different j. Here, a boundary-related ratio is defined as the ratio of the e'_{ij} to its corresponding e_j. A constraint function for particle X_i that is equal to the corresponding threshold indicates that the particle is located on the boundary of the constraint. Accordingly, a particle that uses the ratio to update its position may be moved rapidly to a boundary when optimizing the truss design.

Based on the above heuristic, the goal of boundary-shifting is to force particles to move towards a boundary of a constraint function to reduce the overall search time. The strategy depends on a boundary-shifting function $BS(X_i)$, which is utilized to guide the motion of particles. The function is defined as follows:

$$\begin{aligned} BS(X_i) &= \oplus\sqrt{\min_{j=1}^{J}(|R_{ij}|-1)^2} \quad i=1,2,\ldots,S, j=1,2,\ldots,J \\ \oplus &= \begin{cases} +1 & \text{if } |R_{ij}|>1 \\ -1 & \text{if } |R_{ij}|\le 1 \end{cases} \end{aligned} \tag{3}$$

where $|\cdot|$ and $\min(\cdot)$ denote the absolute and minimum functions, respectively. The aforementioned boundary-related ratio R_{ij} for the ith particle to the jth constraint function, $h_j(X_i)$, is given by $R_{ij} = e'_{ij}/e_j$. The operator $\oplus$ is a sign function dependent on the absolute value of the boundary-related ratio, that is a specific element satisfied $\min(\cdot)$ function. If $|R_{ij}|$ exceeds one, the operator represents $+1$. Otherwise, the operator denotes -1. This study uses the boundary-related ratio to resolve the distance between a corresponding boundary and a particle. If particle X_i is on the boundary of the constraint function, nodal displacement or member stress may equal its predefined constraint threshold, and the boundary-related ratio is unity. Accordingly, the value of $BS(X_i)$ is zero, and the position of the particle need not be adjusted. Otherwise, the new particle position is located by applying the following adjustment formula:

$$X'_i = X_i(1 + BS(X_i)) \tag{4}$$

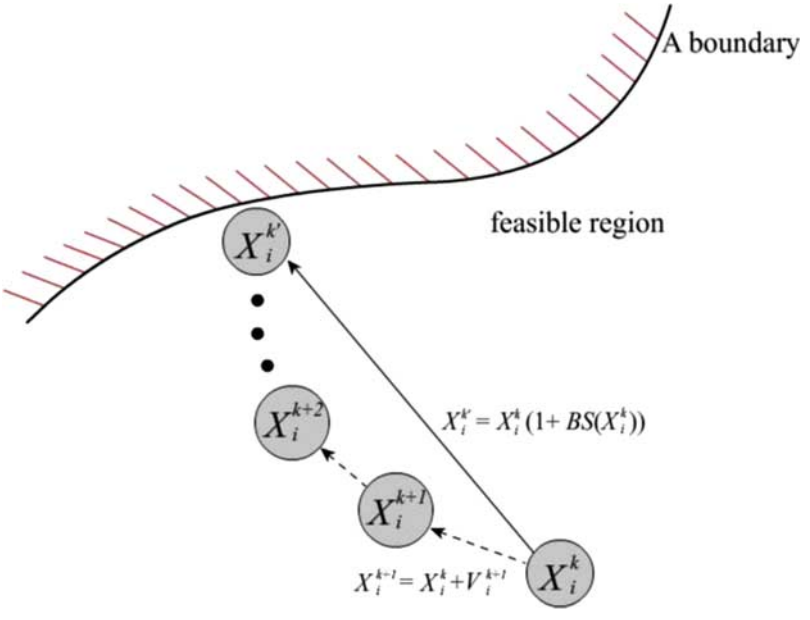

Figure 1. Simple particle search and boundary-shifting search.

A $BS(X_i)$ that exceeds zero implies that the particle X_i violates one or more constraints and that the cross-sectional area of new particle X_i' should be appropriately enlarged. Therefore, the new particle X_i' is moved to the corresponding boundary.

Figure 1 graphically presents the boundary-shifting procedure. Particle X_i^k is moved using the aforementioned procedure so that it approaches the constraint boundary within a few iterations. In the early iterations, system efficiency is enhanced by increasing the convergence rate and reducing the search time. Over several iterations, this strategy moves particles to positions near boundaries so that they converge near the optimal solution. Doing so can diminish the functionality of this strategy and waste processing time in later searches. Accordingly, a threshold value, *e.g.* L_t, must be specified to limit boundary-shifting iterations, which avoids this problem. Meanwhile, forcing all the particles towards the boundaries without any randomness will probably lead to ignoring some other good solutions in the internal regions. Herein, a random parameter (ρ_{bs}) has been adopted in the boundary-shifting strategy as a trigger threshold by a trial-and-error approach.

3.3. *Particle-position-resetting strategy*

An important characteristic of the simple PSO algorithm is its high convergence rate. However, the diversity of particle solutions slowly declines during the rapid convergence process, and the final solution may be a local optimum. The proposed particle-position-resetting method, which is based on the concept of mutation in GA, randomly resets the positions of some, but not all, particles far from the convergence centre. The strategy maintains particle diversity and enables particles to find a possible optimal solution between the edges of the specified range and the predefined limits of design variables. The proposed approach differs from that in related works because it increases diversity by randomly resetting the particle position, although this approach and other related works are all motivated by a random search of mutation in a GA.

The presented strategy comprises three steps. First, a specified range between two reference distances must be determined. The two reference distances are affected by the current optimal

particle P_G in the swarm and the limits, x_n^l and x_n^u, of the problem search space in the nth dimension. Second, if a particle falls within a specified range, a variable is randomly generated using a uniform distribution function in the interval [0,1]. The random variable is compared with a predefined trigger threshold to determine whether or not the particle position is reset. Finally, a reset position is randomly generated to force the particle to move to a new position. The reset positions are always in the problem search space and far from the specified range. The particle-position-resetting approach has three main functions that are described as follows.

- Determining two base reference distances, $d_{bu,n}$ and $d_{bl,n}$, in the nth dimension of the search domain
  The objective of particle-position-resetting is to prevent the convergence of particles to a local optimum and to increase their diversity. The following two base reference distances are defined in terms of the predefined upper and lower limits of each design variable:

$$D_{bu,n} = c_b(x_n^u - P_{G,n}), \quad n = 1, 2, \ldots, N \tag{5}$$

$$D_{bl,n} = c_b(P_{G,n} - x_n^l), \quad n = 1, 2, \ldots, N \tag{6}$$

  where $d_{bu,n}$ and $d_{bl,n}$ are the base reference distances and $P_{G,n}$ is the nth dimension component of the current best particle P_G. The terms x_n^u and x_n^l are the upper and lower limits, respectively, in the nth dimension of the problem search space. c_b is a working variable used to alter the base reference distances and is a constant in the interval [0,1]. Therefore, the two edges of the nth dimension of the specified range are given by the following equations:

$$B_{u,n} = P_{G,n} + d_{bu,n}, \quad n = 1, 2, \ldots, N \tag{7}$$

$$B_{l,n} = P_{G,n} + d_{bl,n}, \quad n = 1, 2, \ldots, N \tag{8}$$

  A range can then be specified according to these edges, *e.g.* $[B_{l,n}, B_{u,n}]$.
- Determining trigger threshold ρ_{fa} and position-resetting probability r_3
  The particle-position-resetting approach is not applied to all particles. A trigger threshold ρ_{fa} is predefined as being within the interval [0,1]. Another essential condition for triggering this scheme is that one particle falls into the specified range $[B_{l,n}, B_{u,n}]$. A position-resetting probability r_3 is then randomly generated and compared with trigger threshold ρ_{fa}. If r_3 is less than ρ_{fa}, the scheme is triggered. A resetting position is determined and this particle may be reset to a new position.
- Determining a resetting position, x_n', in the nth dimension of the search domain for a particle that satisfied the previous conditions
  To prevent the particle from crossing the limits of the problem search space, the upper and lower limits must be employed for a resetting position. Meanwhile, the resetting position cannot fall within the specified range. Based on the aforementioned two requirements, the possible ranges of the resetting position can be represented as grey blocks for the nth dimension in Figure 2. The possible ranges of the resetting position are represented by two blocks that are located out of the specified range. The equation for the total length of the possible range is $(1 - c_b)\,(x_n^u - x_n^l)$. The random resetting position in the nth dimension of the search domain is given by:

$$x_n' = \begin{cases} x_n^l + r_{4,n} & \text{if } r_{4,n} \le (B_{l,n} - x_n^l) \\ x_n^l + c_b(x_n^u - x_n^l) + r_{4,n} & \text{if } r_{4,n} > (B_{l,n} - x_n^l) \end{cases} \tag{9}$$

  where $r_{4,n}$ is a random value between zero and $(1 - c_b)(x_n^u - x_n^l)$ in the nth dimension of problem search space. Other parameters are as defined above.

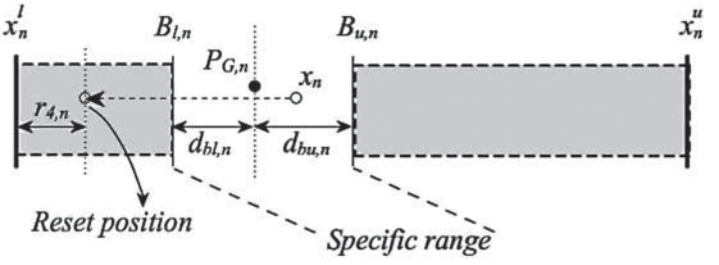

Figure 2. Resetting position and specific range in *n*th dimension of problem space.

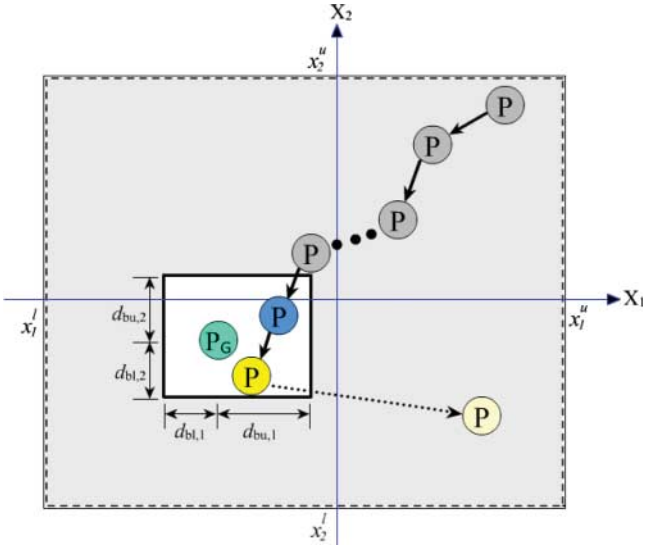

Figure 3. Particle-position-resetting approach in two-dimensional problem space.

Figure 3 graphically presents the particle-position-resetting mechanism in a two-dimensional problem search space. The figure shows a particle P in the problem search space, and position P_G is simultaneously the best current position in the swarm and the centre of convergence. The innermost bold–solid rectangle represents the specified range of base reference distances. The outermost solid rectangle delineates concurrently the problem search space and the boundaries of the resetting position. The solid arrow indicates the searching path, and the dotted arrow indicates the path to the resetting position. When the particle-position-resetting conditions are met, the particle position is reset to the possible ranges between the outermost solid rectangle and the bold–solid rectangle.

3.4. *Constraint handling method*

Penalty functions are often employed to incorporate constraints into the fitness function in a general constrained optimization problem. However, the major disadvantage of penalty functions is that they require moderate tuning to balance the objective function. An approach for handling constraints is the *fly-back* mechanism introduced by He *et al.* (2004a), which is used to maintain a feasible population. Since the population is initialized in the feasible region, flying back to a previous position confirms that the solution is feasible. However, since the proposed approach

only restrains the initial particles in the problem search space, these particles may fall into an infeasible region. The prerequisite for this mechanism is a better particle position falling into a feasible region. The mechanism is then triggered when a particle violates the constraints. The particle is set back to its previous better position and starts a new search in the next iteration. Otherwise, the mechanism is ignored. This mechanism increases the likelihood of the particles exploring the feasible search space near the boundaries.

Herein, the fly-back mechanism is performed in compliance with the boundary shifting strategy after L_t iterations, since the particles may move towards the boundary from the space of feasible solutions or the space of infeasible solutions for the boundary shifting strategy. The shift of moving from an infeasible region to the boundary approximates the mechanism of particle fly-back.

The particles of the violated constraints oscillate in the optimization process. When a particle moves back to a previous location, it can still be stirred into the space of infeasible solutions owing to the normal mechanism of the PSO. However, the global optimum is recorded in each iteration, and changes in the position of the global optimum will disrupt the oscillation. Accordingly, although the particles of the violated constraints oscillate in the optimization process, their oscillation does not influence the best solution, and the best solution can continuously improve.

3.5. *AugPSO algorithm*

The presented AugPSO algorithm integrating a simple PSO and a fly-back mechanism applies two new strategies, boundary-shifting and particle-position-resetting, to increase its convergence rate and to improve the final solution quality. Figure 4 presents the pseudo-code of the AugPSO algorithm. The randomly generated initial positions and velocities for S particles are represented as a swarm. To determine better positions of a particle and the swarm, the objective function of the particle must be calculated. Simultaneously, the constraint functions for the particle are evaluated. By applying a boundary-shifting strategy, the particle can rapidly find a local optimal solution along the boundary of a constraint function, and its effect is significant in the early iterations. Therefore, a threshold, L_t, is used to limit the number of times the strategy is executed in order to avoid waste of computational resources. A random parameter (ρ_{bs}) has been adopted as a probability of boundary-shifting to avoid ignoring some other good solutions in the internal regions. The new and better solutions can be updated by this strategy. The current convergence centre represents the best position of a swarm. The next instance of a particle falling into a specified range may trigger the particle-position-resetting strategy under a probability ρ_{fa}. The particle is randomly reset to the position that is always out of the specified range, and the new and better solution is again updated. The diversity of particle solutions can be increased by the particle-position-resetting strategy. A simple PSO is then started. The new velocity and position of the particle are updated by Equations (1) and (2), respectively. Finally, the fly-back mechanism may be executed depending on the better position of this particle and whether the particle violates a constraint. At the next iteration, the particle is set back to its better position P_i^k. Applying these two new schemes to the PSO improves convergence performance and solution accuracy. The fly-back mechanism also helps the particle to explore the feasible search space near the boundaries.

4. Numerical study

To confirm the feasibility and performance of the proposed new strategies that were adopted in the AugPSO algorithm to optimize the design of truss-type structures, four benchmarked truss design problems were used as test cases. The numerical results achieved using PSO, PSOPC and AugPSO are compared.

```
Randomly generate initial position and velocity
Set all working parameters, MaxIter, c1, c2, Lt, ρfa and cb
k=1
WHILE (terminal conditions are not met) {
    FOR (i=1:S) {
        Calculate objective functions O(Xi)
        Calculate constrained functions hj(Xi)
        //Boundary-shifting strategy
        IF (k<=Lt) {
            IF (randomVlue<= ρbs ) {
                Calculate the boundary-related ratio Rij
                Calculate the boundary-shifting function BS(Xi)
                Update the particle position Xi^k with BS(Xi)
            }
            Re-calculate objective and constraint function of each particle
            Update Pi^k and PG^k
        } //End of boundary-shifting strategy
        //Particle-position-resetting strategy
        IF ( ( Xi^k between Bl,n and Bu,n) and (randomValue<= ρfa ) ) {
            Move Xi^k to random resetting position in outer of [Bl,n ,Bu,n]
            Re-calculate objective and constraint function of each particle
        } //End of particle-position-resetting strategy
        Update Pi^k and PG^k
        Update velocity Vi^k+1 and position Xi^k+1 of all particles
        //Particle fly back
        IF (k>Lt) {
          IF ( Pi^k in the feasible region) and ( Xi^k+1 violate constraint condition) {
            Reset Xi^k+1 to the previous better position Pi^k
          }
        } //End of particle fly back
```

Figure 4. Pseudo-code for the AugPSO algorithm.

In each case of optimization of the design of truss structures under stress and displacement constraints, the elasticity modulus and material density of the structures are specified with the corresponding layout. For all cases a population of 50 particles was utilized. The maximum number of iterations *MaxIter*, used in the proposed AugPSO, was fixed to 400. In four cases, the inertial weight w is set as one and it remains constant over time; the parameters c_1 and c_2 of simple PSO are typically set as two; the limiting iteration L_t and the random parameter ρ_{bs} of the boundary-shifting scheme are set to 20 and 0.9, respectively, by trial and error; the trigger threshold ρ_{fa} and the working variable c_b of particle-position-resetting are set to 0.2 and 0.1, respectively. Table 1 lists the numerical values of the working parameters for the three versions of the PSO in this numerical study. The proposed algorithm was implemented using MATLAB® 7.4 on a PC with an Intel Core™ 2 Quad Processor Q6600 (8 MB cache, 2.40 GHz, 1066 MHz FSB).

4.1. *Ten-bar truss*

The first case concerns a 10-bar truss, which has been considered by several researchers and used as a well-known benchmark problem. Figure 5 presents the layout, structural parameters and loads. The allowable displacement is limited to ±2 in. and the allowable stress is limited to ±25 ksi. The allowable cross-sectional area of all members is between 0.1 in^2 and 35 in^2. To study the contribution of the boundary-shifting and particle-position-resetting scheme to the optimization

Table 1. The working parameters in the numerical study.

Parameters	Algorithms		
	PSO	PSOPC	AugPSO
Inertial weight, w	1	1	1
Cognitive scaling parameters, c_1	2	2	2
Social scaling parameters, c_2	2	2	2
Passive congregation coefficient	None	0.6	None
Limiting iteration, L_t	None	None	20
Trigger threshold, ρ_{fa}	None	None	0.2
Working variable, c_b	None	None	0.1
Boundary shifting random parameter, ρ_{bs}	None	None	0.9

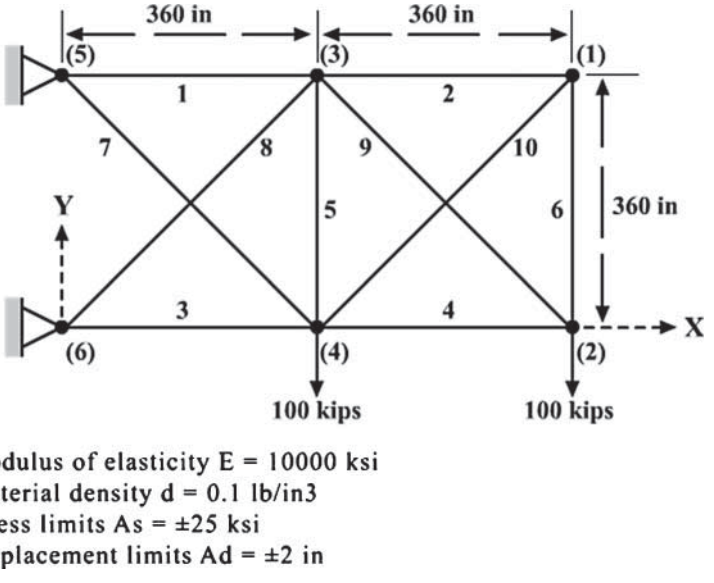

Figure 5. Ten-bar truss with material properties, constraint conditions and loads.

process, an experiment was performed in which one scheme was implemented and compared with a simple PSO algorithm and an AugPSO algorithm. Figure 6 presents the convergence rates of four algorithms: a simple PSO, a simple PSO with the boundary-shifting scheme, a simple PSO with the particle-position-resetting scheme, and the AugPSO. The figure reveals that the boundary-shifting scheme increases the convergence rate in the early stages, and the particle-position-resetting scheme exhibits random progress with similar mutations to those of theGA. As expected, the AugPSO algorithm outperforms the algorithm that was implemented with a single scheme; therefore, follow-up experiments will be performed to compare the AugPSO algorithm with the simple PSO algorithm, PSOPC. Table 2 presents the optimal solutions of the three algorithms and others. The minimum weights of others are better than this work. However, according to the results of Li *et al.* (2007), node 1 violated the displacement constraint. The same result was obtained by Kaveh and Talatahari (2009b). Figure 7 compares the convergence rates of the three algorithms. The AugPSO algorithm has a faster convergence than the PSOPC algorithm in this example, and the solution found using the AugPSO algorithm is superior to that obtained using other algorithms.

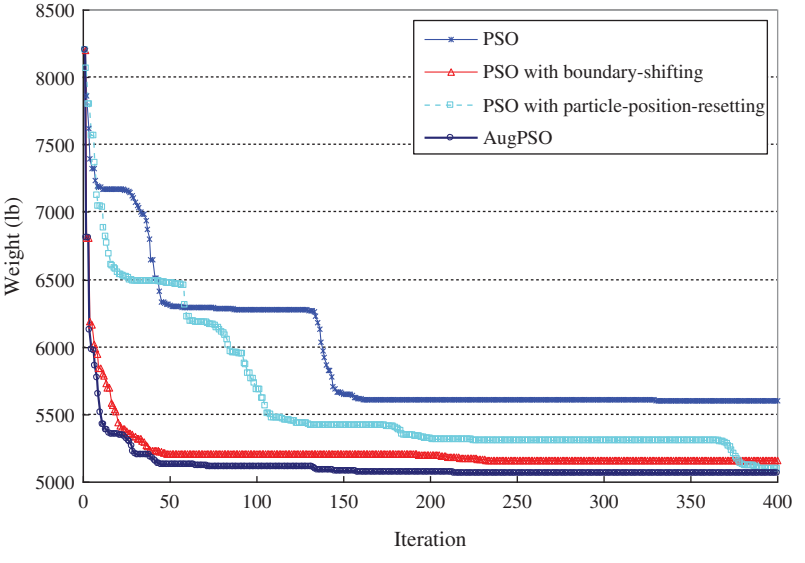

Figure 6. Comparison of convergence rates of simple PSO, PSO with boundary-shifting, PSO with particle-position-resetting and AugPSO algorithm for the 10-bar truss.

Table 2. Optimal solution of the 10-bar truss found using the three algorithms.

Area (in^2)	Li et al. (2007)			Kaveh and Talatahari (2009a)	This work		
	PSO	PSOPC	HPSO	HPSACO	PSO	PSOPC	AugPSO
A1	33.469	30.569	30.704	30.307	20.149	25.923	30.457
A2	0.11	0.1	0.1	0.1	0.1	0.39	0.1
A3	23.177	22.974	23.167	23.434	32.233	23.247	23.584
A4	15.475	15.148	15.183	15.505	14.831	18.208	15.029
A5	3.649	0.1	0.1	0.1	0.1	0.108	0.1
A6	0.116	0.547	0.551	0.5241	0.116	0.1	0.564
A7	8.328	7.493	7.46	7.4365	8.349	9.007	7.42
A8	23.34	21.159	20.978	21.079	28.039	26.629	20.987
A9	23.014	21.556	21.508	21.229	22.909	18.736	21.524
A10	0.19	0.1	0.1	0.1	3.066	0.196	0.1
Weight (lb)	5529.50	5061.00	5060.92[a]	5056.56[a]	5606.04	5225.28	5061.21

Note: [a]Violates constraint.

Herein, the working parameters, L_t, ρ_{bs}, ρ_{fa} and c_b are determined by a trial-and-error approach. Sensitivities to the optimization process were evaluated using 30 runs for this case with the same initials for different values of these parameters. The standard deviations and average weights for each parameter with different values were computed. Analytical results indicate that working parameter L_t equals 20, subsequently yielding the smallest standard deviation and average weight. The random parameter ρ_{bs} equals 0.9, yielding the smallest standard deviation. Depending on the standard deviation and average weight of every 30 runs, the value of trigger threshold ρ_{fa} is set to 0.2. Finally, 30 runs with the same initials for a different value of the C_b value in 400 iterations with 50 particles in a swarm were executed. The number of particle position resetting was counted in each run. In this study, the particle position resetting is a mutation-like operator. For $C_b = 0.1$, the mean percentage of particles whose positions are reset is approximately 4.6%. This value represents a slightly high mutation rate compared with a low mutation probability (inversely proportional to the population size) proposed by De Jong's study (Goldberg 1989). Namely, this

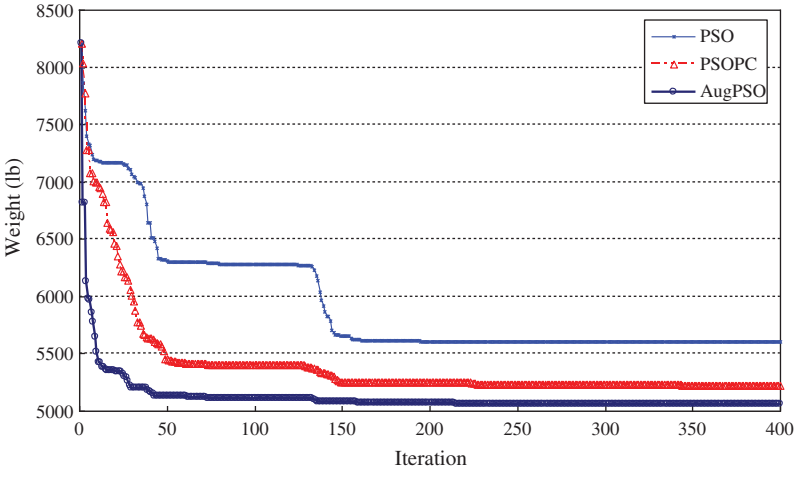

Figure 7. Comparison of convergence rates of the three algorithms for the 10-bar truss.

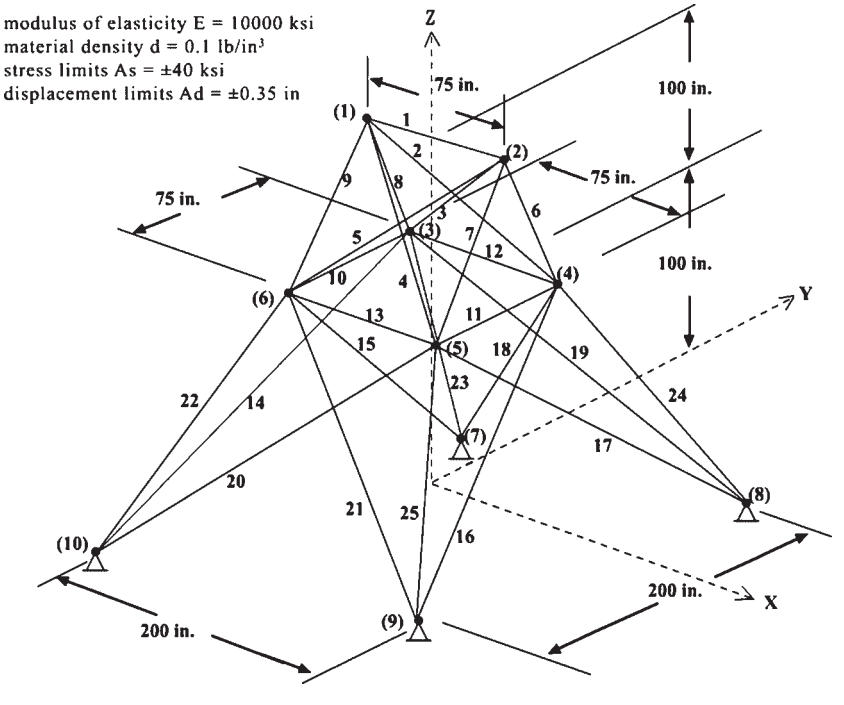

Figure 8. A 25-bar truss with material properties and constraint conditions.

value causes a high probability of triggering particle-position-resetting to increase the diversity of particles.

4.2. *Twenty-five-bar truss*

Figure 8 presents the geometry and design parameters of a 25-bar space truss, considered by many researchers. The allowable displacement is limited to ±0.35 in. and the allowable stress

Table 3. Loads on the 25-bar truss.

	Case I (kips)			Case II (kips)		
Node	P_X	P_Y	P_Z	P_X	P_Y	P_Z
1	0	20	−5	1	10	−5
2	0	−20	−5	0	10	−5
3	0	0	0	0.5	0	0
6	0	0	0	0.5	0	0

Table 4. Optimal solution of the 25-bar truss found using the three algorithms.

Area	Li et al. (2007)			Kaveh and Talatahari (2009a)	This work		
(in^2)	PSO	PSOPC	HPSO	HPSACO	PSO	PSOPC	AugPSO
A1	9.863	0.01	0.01	0.01	0.024	0.013	0.01
A2	1.798	1.979	1.97	2.054	2.246	1.943	1.99
A3	3.654	3.011	3.016	3.008	2.405	3.04	2.989
A4	0.1	0.1	0.01	0.01	0.036	0.011	0.01
A5	0.1	0.1	0.01	0.01	0.01	0.011	0.01
A6	0.596	0.657	0.694	0.679	0.643	0.698	0.68
A7	1.659	1.678	1.681	1.611	1.832	1.695	1.677
A8	2.612	2.693	2.643	2.678	2.875	2.628	2.66
Weight (lb)	627.08	545.27	545.19	544.99[a]	553.627	545.34	545.17

Note: [a]Violates constraint.

is limited to ±40 ksi. Two load cases are given in Table 3, involving stress and displacement constraints. The allowable cross-sectional area of all members is between 0.01 in^2 and 3.4 in^2. The members of this truss are divided into eight groups. Table 4 presents the optimal solutions of the three algorithms and others. The optimal weight that obtained by Kaveh and Talatahari (2009a) was 544.99 lb, which is less than the value obtained in this paper. However, the numerical result concerning the optimal design of Kaveh and Talatahari to operate structural analysis in the SAP2000 reveals that the maximum displacement violates the constraint. For load case 1, the

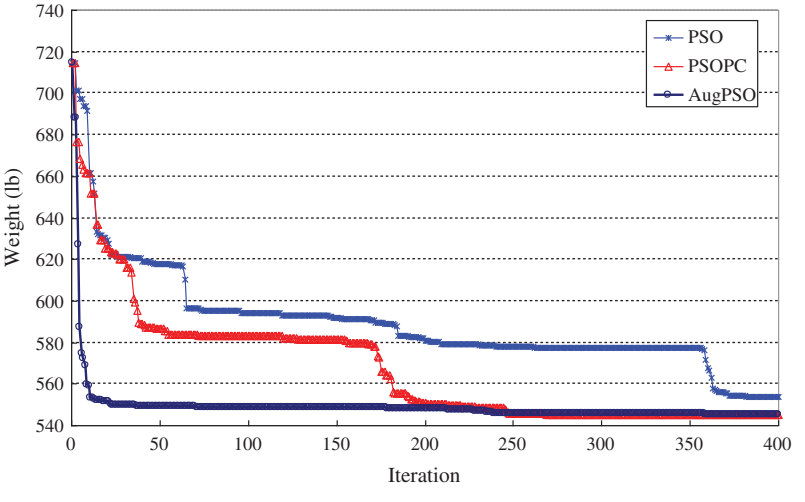

Figure 9. Comparison of convergence rates of the three algorithms for the 25-bar truss.

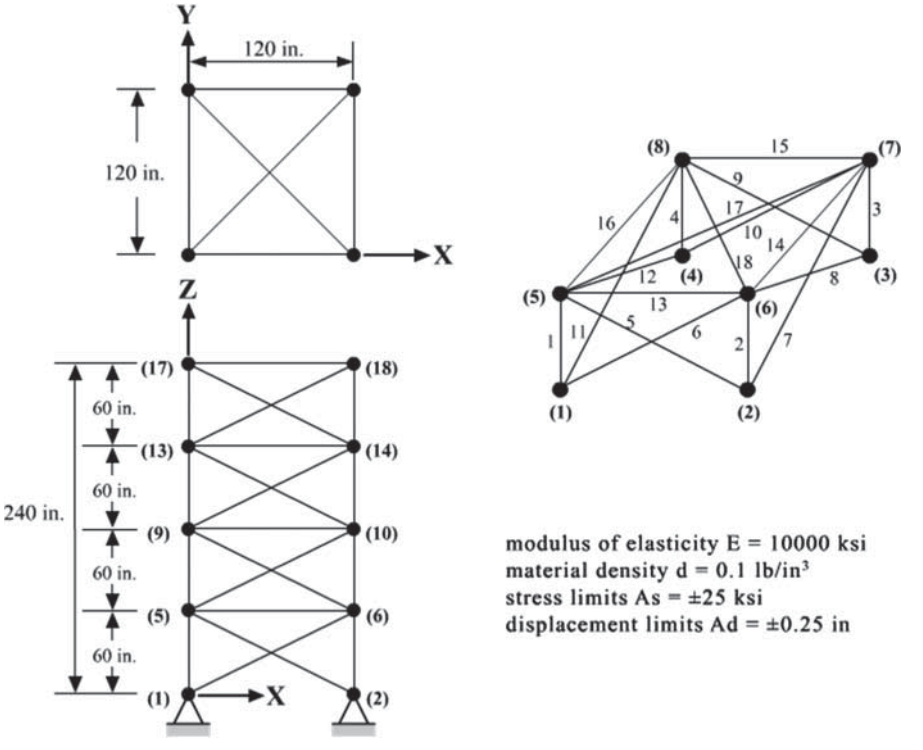

Figure 10. A 72-bar truss with material properties and constraint conditions.

displacement of nodes 1 and 2 in the *Y* direction is 0.35003 and 0.35003 in., respectively. For load case 2, the displacement of nodes 1 and 2 in the *Y* direction is 0.35004 and 0.35004 in., respectively. The results have slightly violated the displacement constraints. The optimal solution of the study is better than Li *et al.* (2007). Figure 9 compares the convergence rates of PSO and PSOPC, and its convergence trend is similar to the 10-bar truss. The convergence rate of AugPSO is better than others and the final solution has the same tendency.

4.3. *Seventy-two-bar truss*

Figure 10 displays the geometry and design parameters of a 72-bar space truss. Li *et al.* (2007) and Perez and Behdinan (2007) optimized the size of the truss using the PSO. The allowable displacement is limited to ±0.25 in. and the allowable stress is limited to ±25 ksi. Two load cases that are given in Table 5 are subjected to stress and displacement constraints. The allowable cross-sectional area of all members is between 0.1 in^2 and 4 in^2. The members of this truss

Table 5. Loads for the 72-bar truss.

	Case I (kips)			Case II (kips)		
Node	P_X	P_Y	P_Z	P_X	P_Y	P_Z
17	5	5	−5	0	0	−5
18	0	0	0	0	0	−5
19	0	0	0	0	0	−5
20	0	0	0	0	0	−5

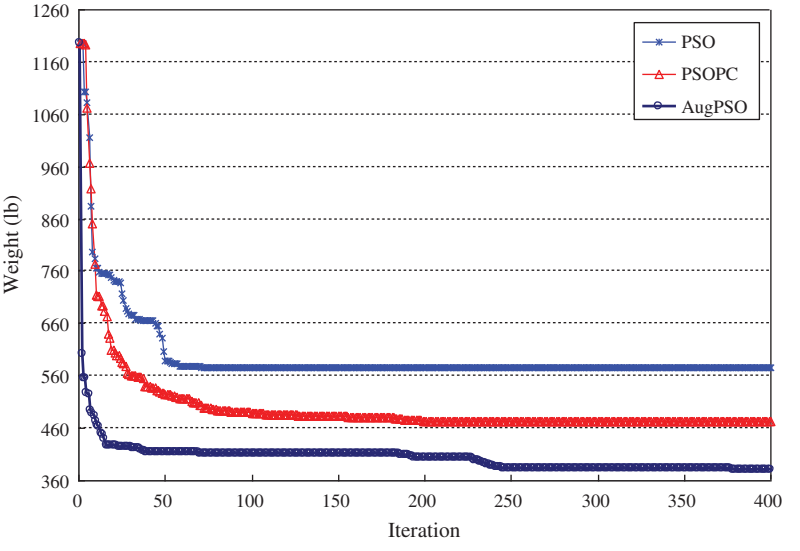

Figure 11. Comparison of convergence rates of the three algorithms for the 72-bar truss.

Table 6. Optimal solution of the 72-bar truss found using the three algorithms.

Area (in^2)	Li et al. (2007)			Perez and Behdinan (2007)	This work		
	PSO	PSOPC	HPSO	PSO	PSO	PSOPC	AugPSO
A1 (1–4)	41.794	1.855	1.857	0.1615	1.609	1.239	1.843
A2 (5–12)	0.195	0.504	0.505	0.5092	0.515	0.513	0.5
A3 (13–16)	10.797	0.1	0.1	0.4967	0.888	0.100	0.104
A4 (17–18)	6.861	0.1	0.1	0.5619	1.513	0.426	0.1
A5 (19–22)	0.438	1.253	1.255	0.5142	1.003	1.302	1.221
A6 (23–30)	0.286	0.505	0.503	0.5464	0.382	0.740	0.549
A7 (31–34)	18.309	0.1	0.1	0.1	0.100	0.100	0.1
A8 (35–36)	1.22	0.1	0.1	0.1095	1.390	1.896	0.1
A9 (37–40)	5.933	0.497	0.496	1.3079	0.560	0.491	0.49
A10 (41–48)	19.545	0.508	0.506	0.5193	0.668	0.558	0.496
A11 (49–52)	0.159	0.1	0.1	0.1	0.220	0.100	0.103
A12 (53–54)	0.151	0.1	0.1	0.1	1.842	0.127	0.15
A13 (55–58)	10.127	0.1	0.1	1.7427	0.120	0.157	0.156
A14 (59–66)	7.32	0.525	0.524	0.5185	0.665	0.490	0.575
A15 (67–70)	3.812	0.394	0.4	0.1	0.527	0.404	0.433
A16 (71–72)	18.196	0.535	0.534	0.1	0.413	0.950	0.522
Weight (lb)	6818.67	369.65[a]	369.65[a]	381.91	576.694	472.586	381.62

Note: [a]Violates constraint.

are grouped into 16 categories. Figure 11 compares the convergence rates of PSO and PSOPC and Table 6 presents the optimal solution. The optimal truss weight that is determined using AugPSO is 381.62 lb. The weight is better than the result of Perez and Behdinan (2007). The maximum values of displacements of node 17 in the X and Y directions are 0.25 in., satisfying the constraint in load case 1. Under load case 2, the maximum displacement of 17–20 in the Z direction satisfies the displacement constraint. The extreme values of stress in the two load cases are −16.1115 and −24.6374 ksi, respectively. These values are close to the allowable stress and do not exceed the limit value. The optimal weight obtained by Li *et al.* (2007) was 369.65 lb, which is less than the value obtained in this study. However, the numerical result concerning the

optimal design of Li *et al.* to operate structural analysis in the SAP2000 reveals that the maximum displacement violates the constraint. The displacements of nodes 17–20 in the Z direction violate the displacement constraint in load case 2, and the stresses of members 55–58 (the pillars on the top floor) violate the stress constraint. The result of AugPSO satisfies the constraints and exhibits powerful search ability.

4.4. *Dome truss with 120 bars*

Figure 12 shows the geometry and design parameters of a 120-bar dome truss with 49 nodes. The limiting displacement is ±0.1969 in. along three perpendicular axes and the yield stress F_y

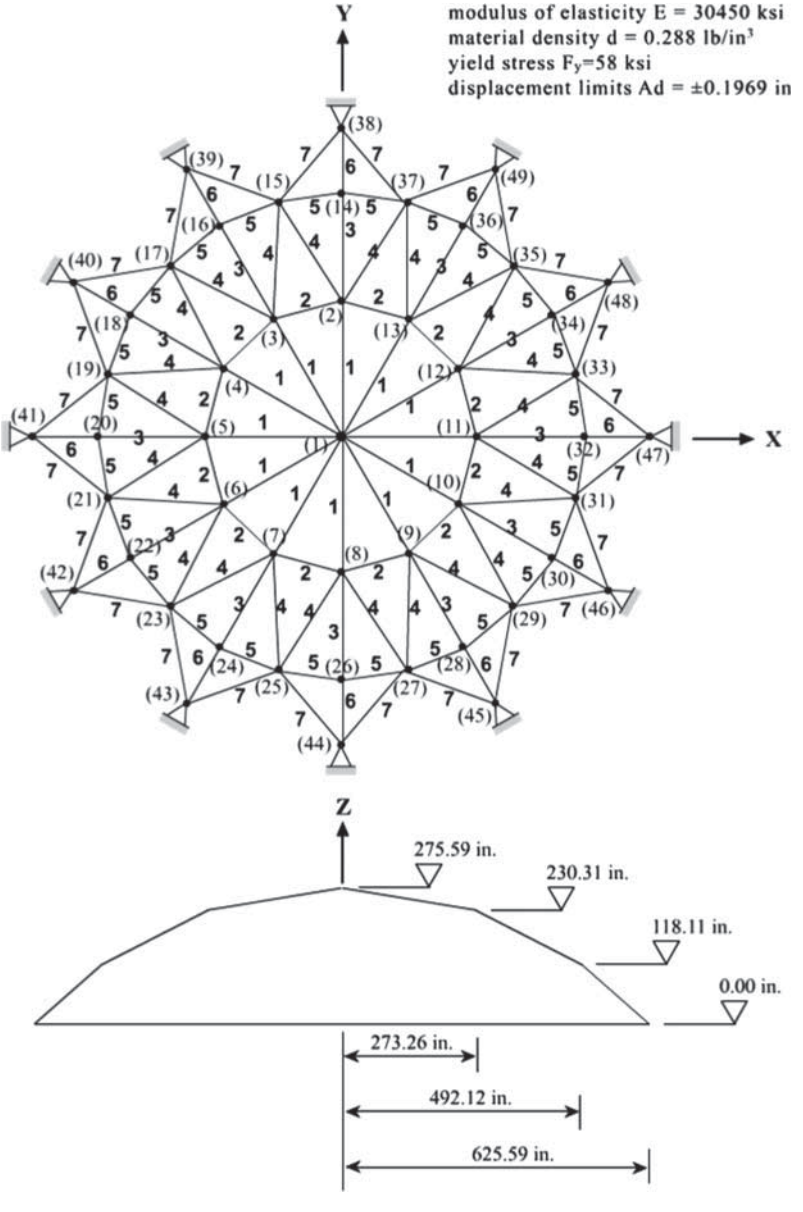

Figure 12. A 120-bar dome truss with material properties and constraint conditions.

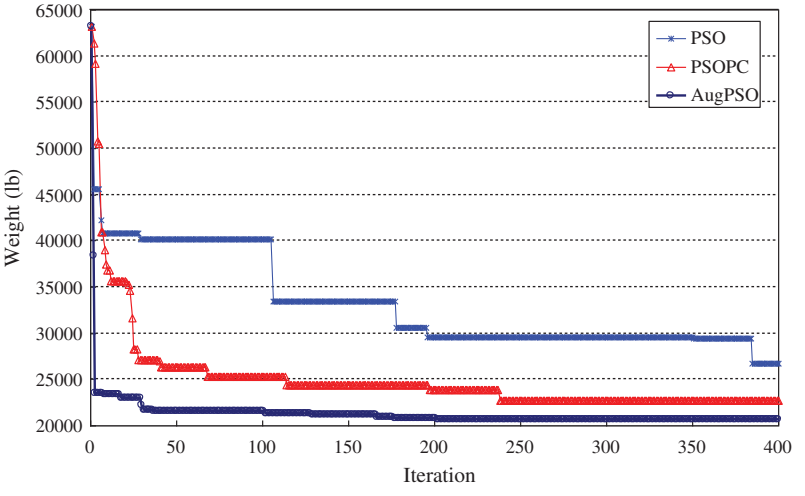

Figure 13. Comparison of convergence rates of the three algorithms for the 120-bar truss.

is 58 ksi. The allowable cross-sectional area of all members is between 0.775 in^2 and 20 in^2. The members of this dome truss are divided into seven groups. In addition, the allowable tensile and compressive stresses are used in a manner consistent with the AISC Allowable Stress Design for Steel Structures Code in the following equation:

$$\begin{cases} \sigma_i^+ = 0.6F_y & \text{for } \sigma_i \geq 0 \\ \sigma_i^- & \text{for } \sigma_i < 0 \end{cases} \tag{10}$$

where allowable compressive stress σ_i^- is calculated using the following equation:

$$\sigma_i^- = \begin{cases} \left[\left(1 - \frac{\lambda_i^2}{2C_c^2}\right)F_y\right] \Big/ \left(\frac{3}{5} + \frac{3\lambda_i}{8C_c} - \frac{\lambda_i^3}{8C_c^3}\right) & \text{for } \lambda_i < C_c \\ \frac{12\pi^2 E}{23\lambda_i^2} \quad \text{for } \lambda_i \geq C_c \end{cases} \tag{11}$$

where E is the modulus of elasticity; F_y denotes the yield stress of steel; C_c is $\sqrt{2\pi^2 E/F_y}$; λ_i is the slenderness ratio (kL_i/r_i); and k, L_i and r_i represent the effective length factor, the member length and the radius of gyration, respectively.

The dome truss is subjected to vertical loading at its unsupported joints in each layer ring. These loads were taken to be −13.49 kips at node 1, −6.744 kips at nodes 2 through 14, and −2.248 kips at other unsupported nodes. Figure 13 compares the convergence rates with those of PSO and PSOPC and Table 7 presents the optimal solution. Notably, the optimal weight determined using the AugPSO is 20,675.545 lb in this case, and the result is outstanding relative to other results (Kelesoglu and Ulker 2005, Kaveh and Talatahari 2008, 2009a) in the literature. The results were reanalysed using SAP2000 and the solutions were confirmed to be feasible.

4.5. *Numerical comparison*

In addition to comparing optimal solutions and convergence rates by numerical study, the standard deviation and average weight of every 30 runs were summarized to compare the robustness of the AugPSO with those of the simple PSO and PSOPC algorithms. The same initials and working parameters were used in 30 runs of each algorithm. Table 8 lists the numerical results. The best, worst and average weights of truss structure and standard deviation were evaluated in four cases.

Table 7. Optimal solution of the 120-bar dome found using the three algorithms.

Area (in^2)	Kaveh and Talatahari (2009a)			This work		
	PSO	PSOPC	HPSACO	PSO	PSOPC	AugPSO
A1	12.802	3.04	3.095	3.331	3.325	3.287
A2	11.765	13.149	14.405	5.397	5.972	3.486
A3	5.654	5.646	5.02	4.513	4.505	4.256
A4	6.333	3.143	3.352	3.272	3.026	2.752
A5	6.963	8.759	8.631	1.674	0.914	1.353
A6	6.492	3.758	3.432	6.459	4.153	3.507
A7	4.988	2.502	2.499	3.548	2.415	2.411
Weight (lb)	51,986.2	33,481.2	33,248.9	26,729.0	22,654.3	20,675.5

Table 8. Performance comparison of PSO-based algorithms for all trusses in 30 runs.

Members	Algorithm	Best weight (lb)	Worst weight (lb)	Average weight (lb)	Standard deviation (lb)
10	PSO	5,606.036	7,008.815	6,256.895	370.071
	PSOPC	5,225.282	5,854.163	5,425.074	175.764
	AugPSO	5,061.209	5,179.516	5,103.484	31.755
25	PSO	553.627	662.835	597.925	23.970
	PSOPC	545.388	594.448	556.812	13.357
	AugPSO	545.173	576.190	551.864	7.677
72	PSO	576.694	789.080	662.093	60.454
	PSOPC	472.586	643.165	545.376	47.225
	AugPSO	381.616	436.914	414.044	12.282
120	PSO	26,728.980	53,777.510	31,401.182	5653.241
	PSOPC	22,654.325	38,471.212	26,424.703	4103.610
	AugPSO	20,675.545	21,678.621	21,175.514	259.105

For the 10-bar truss, the best weights obtained by AugPSO, PSOPC and PSO are 5061.209, 5225.282 and 5606.036 lb, and the standard deviations are 31.755, 175.764 and 370.071 lb, respectively. In all cases, the AugPSO outperformed the PSO and PSOPC in terms of best and average weights. The AugPSO also outperformed the PSO and PSOPC in terms of its standard deviation. The results show that the AugPSO is highly stable in four cases and is more accurate and robust than PSO and PSOPC.

5. Concluding remarks

This work proposed an augmented particle swarm optimization (AugPSO) algorithm that integrates two new strategies, a heuristic-inspired boundary-shifting approach and a mutation-like particle-position-resetting strategy, to accelerate convergence and to ensure that the solution does not fall into a local optimum. Introducing the boundary-shifting strategy enables particles to be located close to the constraint boundaries so that the optimal solution can be found quickly. The required time to implement the strategy in the early stage of the optimization procedure is also reduced. The particle-position-resetting strategy maintains particle diversity in the fast convergence process and ensures that the final solution is not a local optimum.

Four numerical results of truss structure optimization were used to test the AugPSO algorithm. The proposed algorithm yields higher quality solutions and higher convergence rates than PSO and PSOPC. The efficiency of the AugPSO algorithm is evident in the need for fewer iterations to achieve the goal. In each test case, the AugPSO algorithm converged to a better solution in fewer

than 50 iterations. The final solution quality achieved by the AugPSO is comparable to those achieved by PSO and PSOPC. Notably, the optimal solution for the 120-bar dome truss is much better than several other algorithms reported in the literature (Kelesoglu and Ulker 2005, Kaveh and Talatahari 2008, 2009a). The AugPSO provided a superior optimal solution of 20,675.545 lb.

In addition to evaluating only the Student's *t*-test of the average weight between algorithms, this work compared the mean of the weights obtained in 30 runs for each algorithm. The results reveal that the average weight of AugPSO is smaller than that of PSO and PSOPC. Meanwhile, comparisons of best weights also show that the AugPSO is more robust than PSO and PSOPC in all cases. Owing to the superior functionality achieved by the two new strategies introduced in this study, the presented AugPSO algorithm appears more robust than other algorithms.

References

Andrews, P.S., 2006. An investigation into mutation operators for particle swarm optimization. *In*: *2006 IEEE congress on evolutionary computation (CEC2006)*, 16–21 July, Vancouver, Canada. Piscataway, NJ: IEEE Press, 1044–1051.

Binkley, K.J. and Hagiwara, M., 2008. Balancing exploitation and exploration in particle swarm optimization: velocity-based reinitialization. *Information and Media Technologies*, 3, 103–111.

Charles, V.C. and Barron, J.B., 2004. Design of space trusses using ant colony optimization. *Journal of Structural Engineering – ASCE*, 130, 741–751.

Clerc, M., 1999. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. *In*: *1999 IEEE congress on evolutionary computation*, 6–9 July Washington, DC. New York: IEEE Press, 1951–1957.

Eberhart, R. and Shi, Y., 2000. Comparing inertia weights and constriction factors in particle swarm optimization. *In*: *2000 IEEE Congress on Evolutionary Computation*, 16–19 July La Jolla, CA. Washington, D.C.: IEEE Computer Society Press, 84–88.

Engelbrecht, A.P., 2005. *Fundamentals of computational swarm intelligence*. 1st ed. New Jersey: Wiley.

Fan, S.-K.S. and Chiu, Y.Y., 2007. A decreasing inertia weight particle swarm optimizer. *Engineering Optimization*, 39, 203–228.

Fan, S.-K.S. and Chang, J.M., 2009. A parallel particle swarm optimization algorithm for multi-objective optimization problems. *Engineering Optimization*, 41, 673–697.

Fan, S.-K.S. and Chang, J.M., 2010. Dynamic multi-swarm particle swarm optimizer using parallel PC cluster systems for global optimization of large-scale multimodal functions. *Engineering Optimization*, 42, 431–451.

Fourie, P.C. and Groenwold, A.A., 2002. The particle swarm optimization algorithm in size and shape optimization. *Structural and Multidisciplinary Optimization,* 23, 259–267.

Goldberg, D.E., 1989. *Genetic algorithms in search, optimization and machine learning*. 1st ed. Boston, MA: Addison-Wesley.

He, S., *et al.*, 2004a. An improved particle swarm optimizer for mechanical design optimization problems. *Engineering Optimization*, 36, 585–605.

He, S., *et al.*, 2004b. A particle swarm optimizer with passive congregation. *Biosystems*, 78, 135–147.

Heppner, F. and Grenander, U., 1990. A stochastic nonlinear model for coordinated bird flocks. *In:* S. Krasner, ed. *The ubiquity of chaos*. Washington, DC: AAAS Publications, 233–238.

Kaveh, A. and Shojaee, S., 2007. Optimal design of skeletal structures using ant colony optimization. *International Journal for Numerical Methods in Engineering*, 70, 563–581.

Kaveh, A. and Talatahari, S., 2008. A hybrid particle swarm and ant colony optimization for design of truss structures. *Asian Journal of Civil Engineering (Building and Housing)*, 9, 329–348.

Kaveh, A. and Talatahari, S., 2009a. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Computer & Structure*, 87, 267–283.

Kaveh, A. and Talatahari, S., 2009b. A particle swarm ant colony optimization for truss structures with discrete variables. *Journal of Constructional Steel Research*, 65, 1558–1568.

Kelesoglu, O. and Ulker, M., 2005. Fuzzy optimization of geometrical nonlinear space truss design. *Turkish Journal of Engineering & Environmental Sciences*, 29, 321–329.

Kennedy, J. and Eberhart, R., 1995. Particle swarm optimization. *In*: *Proceedings of IEEE international conference on neural networks*, 27 November–1 December Perth, 4. Piscataway, NJ: IEEE Press, 1942–1948.

Kennedy, J., *et al.*, 2001. *Swarm intelligence*. San Francisco: Morgan Kaufmann.

Li, L.J., *et al.*, 2007. A heuristic particle swarm optimizer for optimization of pin connected structures. *Computer & Structure*, 85, 340–349.

Li, L.J., *et al.*, 2006. An improved particle swarm optimization method and its application in civil engineering. *In*: *Proceedings of the 5th international conference on engineering computational technology*, 12–15 September Las Palmas de Gran Canaria. Stirlingshire: Civil-Comp Press, Paper 42.

Liu, F., *et al.*, 2006. An improved particle swarm optimization method and its application in steel structures. *In*: *International symposium on new Olympic, new shell and spatial structures*, 16–19 October, Beijing, China. Beijing University of Technology Press, 282–283.

Ling, S.H., *et al.*, 2008. Hybrid particle swarm optimization with wavelet mutation and its industrial applications. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 38, 743–763.

Montazeri-Gh, M., *et al.*, 2012. Application of particle swarm optimization in gas turbine engine fuel controller gain tuning. *Engineering Optimization*, 44, 225–240.

Perez, R.E. and Behdinan, K., 2007. Particle swarm approach for structural design optimization. *Computer & Structure*, 85, 1579–1588.

Rao, S. S., 2009. *Engineering optimization: theory and practice*. 4th ed. New Jersey: Wiley.

Reynolds, C.W., 1987, Flocks, herds, and schools: a distributed behavioral model. *Computer Graphics*, 21 (4), 25–34.

Schutte, J.F. and Groenwold, A.A., 2003. Sizing design of truss structures using particle swarms. *Structural and Multidisciplinary Optimization*, 25, 261–269.

Shelokar, P.S., *et al.*, 2007. Particle swarm and ant colony algorithms hybridized for improved continuous optimization. *Applied Mathematics and Computation*, 188, 129–142.

Shi, Y. and Eberhart, R., 1998. A modified particle swarm optimizer. *In: 1998 IEEE International Conference on Evolutionary Computation proceedings*, 4–9 May, Anchorage, Alaska. Piscataway, NJ: IEEE Press, 69–73.

Singh, H.K., *et al.*, 2009. Performance of infeasibility driven evolutionary algorithm (IDEA) on constrained dynamic single objective optimization problems. *In*: *2009 IEEE congress on evolutionary computation*, 18–21 May, Trondheim, Norway. Piscataway, NJ: IEEE Press.

Stacey, A., *et al.*, 2003. Particle swarm optimization with mutation. *In*: *2003 IEEE congress on evolutionary computation*, 8-12 December, Canberra, Australia. Los Alamitos: IEEE Press, 1425–1430.

Wang, Y.X., *et al.*, 2010. Particle swarm optimizer with adaptive tabu and mutation: a unified framework for efficient mutation operators. *ACM Transactions on Autonomous and Adaptive Systems*, 5 (1), Article 1.

Wu, S.J. and Chow, P.T., 1995. Steady-state genetic algorithms for discrete optimization of trusses. *Computer & Structure*, 56, 979–991.