Vol. 00, No. 00, October 2008, 1–29

RESEARCH ARTICLE

Solving Constrained Optimization Problems with a Hybrid Particle Swarm Optimization Algorithm

Leticia Cecilia Cagnina, Susana Cecilia Esquivel[1]
and Carlos A. Coello Coello[2]*
[1]*LIDIC (Research Group). Universidad Nacional de San Luis*
Ej. de Los Andes 950. (D5700HHW) San Luis, ARGENTINA
[2]*CINVESTAV-IPN (Evolutionary Computation Group)*
Departamento de Computación, Av. IPN No. 2508
Col. San Pedro Zacatenco, México D.F. 07360, MEXICO
Email: {lcagnina,esquivel}@unsl.edu.ar *and* ccoello@cs.cinvestav.mx

(*v3.7 released September 2008*)

This paper presents a particle swarm optimization algorithm for solving general constrained optimization problems. The proposed approach introduces different methods to update the particle's information, as well as the use of a double population and a special shake mechanism designed to avoid premature convergence. It also incorporates a simple constraint-handling technique. Twenty-four constrained optimization problems commonly adopted in the evolutionary optimization literature, as well as some structural optimization problems are adopted to validate the proposed approach. The results obtained by the proposed approach are compared with respect to those generated by algorithms representative of the state of the art in the area.

Keywords: particle swarm optimization; constraint-handling; evolutionary algorithms; engineering optimization

*Corresponding author

ISSN: 0305-215X print/ISSN 1029-0273 online
© 2008 Taylor & Francis
DOI: 10.1080/0305215YYxxxxxxxx
http://www.informaworld.com

1. Introduction

In recent years, a wide variety of real-world applications have been solved using metaheuristics, mainly because they tend to produce reasonably good solutions at a reasonable computational cost. One of the metaheuristics that has become very popular in the last few years is Particle Swarm Optimization (PSO) (Kennedy and Eberhart 2001).

PSO was conceived as a simulation of individual and social behavior (Kennedy and Eberhart 1999) such as the one observed in flocks of birds and fishes. PSO explores the search space using a population of individuals (named *swarm*), and the best performers (either within a group or with respect to the entire population) affect the performance of the others.

In PSO, each individual is named *particle* and represents a possible solution to the problem at hand, within a multidimensional search space. The particles have their own position and velocity and such values are updated at each iteration of the algorithm. The individuals also record their past behavior and use it to move towards promising regions of the search space.

PSO has been found to be highly competitive for solving a wide variety of optimization problems (Bochenek and Forys 2006, Muñoz-Zavala *et al.* 2006, Liang and Suganthan 2006, He and Wang 2007, Ye *et al.* 2007, Perez and Behdinan 2007b, Mezura-Montes and López-Ramírez 2007). In spite of the popularity of PSO as an efficient optimizer, it has been until relatively recently that its use has focused more on engineering optimization problems, mainly because of the lack of constraint-handling techniques that had been explicitly designed to be coupled with a PSO[1] algorithm. Although the use of constraint-handling mechanisms commonly adopted with other evolutionary algorithms is, of course, possible in PSO (e.g., exterior penalty functions), the resulting approaches are normally not very competitive with respect to state-of-the-art evolutionary optimization algorithms, which motivates the development of carefully designed constraint-handling mechanisms that explicitly exploit the features of PSO when exploring the search space. In this paper, it is precisely introduced a proposal of this sort, which can be useful to solve nonlinear constrained optimization problems. The approach is validated first with a test suite commonly adopted in the literature of constrained evolutionary optimization (i.e., evolutionary algorithms that have a constraint-handling mechanism), in order to assess the competitiveness of the present approach with respect to state-of-the-art evolutionary algorithms designed for solving constrained optimization problems. Then, in the final part of the paper, some engineering optimization problems are presented and the results are compared with respect to approaches that have been used to solve them in the specialized literature.

The remainder of the paper is organized as follows. Section 2 describes the general nonlinear optimization problem of interest. A review of the most representative previous related work is presented in Section 3. Section 4 describes the proposed PSO for solving constrained optimization problems. The experimental setup and the comparison of results with respect to state-of-the-art approaches are presented in Section 5. Finally, the conclusions and some possible paths for future work are presented in Section 6.

[1]PSO, like any other evolutionary algorithm (e.g., genetic algorithms) can be seen as an unconstrained search/optimization technique, since in its original form, it does not have an explicit constraint-handling mechanism.

2. Statement of the Problem

The focus of this paper is the solution of the general **nonlinear optimization problem** in which the objective is:

$$\text{Find } \vec{x} \text{ which optimizes } f(\vec{x}) \tag{1}$$

subject to:

$$g_i(\vec{x}) \leq 0, \quad i = 1, \ldots, n \tag{2}$$

$$h_j(\vec{x}) = 0, \quad j = 1, \ldots, p \tag{3}$$

where $\vec{x}$ is the vector of solutions $\vec{x} = [x_1, x_2, \ldots, x_r]^T$, n is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or nonlinear).

If $\mathcal{F}$ denotes the feasible region and $\mathcal{S}$ denotes the whole search space, then it should be clear that $\mathcal{F} \subseteq \mathcal{S}$.

For an inequality constraint that satisfies $g_i(\vec{x}) = 0$, then it is said to be **active** at $\vec{x}$. All equality constraints h_j (regardless of the value of $\vec{x}$ used) are considered active at all points of $\mathcal{F}$.

It is relatively common to transform equality constraints into inequalities of the form:

$$|h_j(\vec{x})| - \epsilon \leq 0 \tag{4}$$

where ϵ is the tolerance allowed (a very small value). This is precisely the approach adopted in this paper. More details about the definition of ϵ are provided in Section 4.2.4.

3. Previous Related Work

As indicated before, evolutionary algorithms can be seen as unconstrained search techniques, since in their original form, they do not incorporate any explicit mechanism to handle constraints. Because of this, several authors have proposed a variety of constraint-handling techniques explicitly designed for evolutionary algorithms (Coello Coello 2002, Mezura-Montes 2009).

However, there exist relatively few proposals involving the design of constraint-handling techniques explicitly developed for the PSO algorithm. Next, a short review of the most representative work done in this regard is shown.

Hu and his co-workers (2002, 2003) proposed mechanisms that ensure the generation of feasible solutions. Such mechanisms can be, however, very expensive (computationally speaking) and even impossible to use in problems having a very small feasible region. The test cases adopted to validate this approach were a few engineering optimization problems in which the size of the feasible region is relatively large with respect to the total size of the search space.

Paquet and Engelbrecht (2003) proposed an approach explicitly designed to deal with linear constraints, but without considering a more general extension that incorporates nonlinear constraints.

Zhang et al. (2004) proposed the use of a *periodic* constraint-handling mode in a PSO algorithm. The main idea is to make periodic copies of the search space when the algorithm starts the run. This aims to avoid the dispersion of particles that arises from the application of the mutation operator to particles lying on the boundary between the feasible and infeasible regions. This approach was validated adopting a low number of objective function evaluations (ranging from 28,000 to 140,000), and using eight test problems. The results produced by the proposed approach were compared with respect to those generated by traditional constraint-handling techniques (i.e., penalty functions), but none is provided with respect to state-of-the-art evolutionary algorithms designed for constrained search spaces.

In Toscano Pulido and Coello Coello (2004) a simple constraint-handling mechanism based on closeness of the particles in the swarm to the feasible region is incorporated into a PSO algorithm. This approach also incorporates a mutation operator (called *turbulence*), which changes the flight of the particles to different zones, aiming to maintain diversity. In the validation of this approach, the authors adopted a relatively large population size, and a low number of iterations, as to perform 340,000 objective function evaluations. The results of this approach were found to be competitive with respect to those generated by state-of-the-art evolutionary algorithms designed for constrained optimization (namely, stochastic ranking (Runarsson and Yao 2000), homomorphous maps (Koziel and Michalewicz 1999) and ASCHEA (Hamida and Schoenauer 2002)) when solving the thirteen test problems adopted in (Runarsson and Yao 2000).

Parsopoulos et al. (2005) proposed a *Unified Particle Swarm Optimization* approach, which was then adapted to incorporate constraints. This approach adopts a penalty function, which uses information from the number of constraints violated and the magnitude of such violations. Also, the feasibility of the best solutions is preserved. This approach was tested with four constrained engineering optimization problems with promising results. However, no results were presented with benchmark problems, which are normally more difficult to solve.

Lu et al. (2006) proposed DOM, a dynamic objective technique to handle constraints, based on the search mechanism of the PSO algorithm. DOM states bi-objective unconstrained optimization problems: one objective is related to reaching the feasible region, and the other is related to reaching the global optimum. The technique allows each particle to dynamically adjust the importance given to each of these two objectives, based on their current position. The authors also presented a restricted velocity PSO (RVPSO) which incorporates information about the feasible region that had been previously learnt. Both approaches were validated using the 13 well-known benchmark functions adopted in (Runarsson and Yao 2000). DOM was compared with an approach called "keeping feasible solutions", which was reported in (Hu and Eberhart 2002) and the former outperformed the latter. Then, they incorporated DOM into RVPSO and also into a CPSO algorithm (i.e., a PSO algorithm using the constriction factor (Eberhart and Shi 2000)), and the results showed that DOM+RVPSO outperformed or exhibited the same performance as DOM+CPSO. The results were also compared with respect to PESO (a PSO approach that incorporates some evolutionary operators that improve its performance) (Muñoz-Zavala *et al.* 2005) and they concluded that their proposed approach was highly competitive when considering the quality of the solutions produced. Additionally, the authors indicated that another advantage was that their proposed approach required only 50,000 objective function evaluations, whereas PESO required 350,000 evaluations. A follow-up of this work was presented in (Lu and Chen 2008), in which the authors proposed an approach called "self-adaptive velocity PSO" (SAVPSO), which is based on DOM. This version uses a different velocity update equation to maintain the particles into the feasible region, and the previous experience incorporated into the velocity is restricted to the flying direction. This version was assessed using the same set of 13 test problems indicated before. Results were compared with respect to those reported by Toscano Pulido and Coello Coello (2004), PESO (Muñoz-Zavala *et al.* 2005), Hu and Eberhart (2002) and DEPSO (a PSO with a reproduction operator similar to the one adopted by differential evolution) (Zhang and Xie 2003).

Liang and Suganthan (2006) modified a previous dynamic multi-swarm particle swarm optimizer (DMS-PSO) to solve 24 benchmark functions. Their version includes a new method to handle constraints based on sub-swarms which are assigned depending on the difficulty of each constraint. The algorithm, named DMS-C-PSO (for DMS-PSO + Constraint mechanism), has dynamic multi-swarms and a Sequential Quadratic Programming method (used as a local search engine) aimed to improve the DMS-C-PSO's ability to find good solutions. The authors tested their approach with 24 constrained test problems. They concluded that their proposed approach was able to find feasible solutions in all the test problems and that was able to reach the optimum in most of them.

Zahara and Hu (2008) proposed a hybridization of PSO with the Nelder-Mead method (Nelder and Mead 1965), called NM-PSO. The aim was to combine the global search properties of PSO with the efficient local search performed by the Nelder-Mead method. This approach adopts two constraint-handling methods: a gradient repair technique and a constraint fitness priority-based ranking method. Both of them avoid the difficulties associated with the choice of an appropriate penalty term within a penalty function, by using gradient information derived from the set of constraints of the problem. This, however, eliminates the main advantage of using a derivative-free search method (Nelder-Mead), since such gradient information will be required and its estimation requires several additional objective function evaluations, unless the exact derivatives are available. NM-PSO was applied to the 13 benchmark problems included in (Runarsson and Yao 2000). The approach performed 5,000 iterations in all cases, but the number of objective function evaluations performed at each iteration depends on the dimensionality of the problem (the population size was set to $21 \times N + 1$, where N is the number of decision variables of the problem). In some cases, the number of objective function evaluations required by this approach was close to one million, which is a very high value when compared to the number of evaluations typically required by modern constraint-handling techniques (normally no higher than 500,000). Results were compared with respect to a cultured differential evolution approach (Landa Becerra and Coello Coello 2006), filter simulated annealing (Hedar and Fukushima 2006), a genetic algorithm (Chootinan and Chen 2006) and an improved version of stochastic ranking (Runarsson and Yao 2005). In terms of the quality of the results achieved, NM-PSO was able to find the best known solutions in 10 problems, while the others could only find the best solutions in eight of them. In terms of computational cost, NM-PSO required less objective function evaluations in eight problems.

Mezura-Montes and Flores-Mendoza (2009) evaluated with 24 constrained functions some PSO variants with the purpose of selecting the most competitive local best PSO with a constriction factor. Then, the authors modified that PSO variant by adding to it two features in order to obtain the so-called Improved Particle Swarm Optimization

(IPSO) algorithm. One modification was the incorporation of a dynamic adaptation mechanism to control two parameters: the acceleration constant that controls the influence of social information (c_2) and the constriction factor (k). The second modification was in the dominance criterion used to compare solutions: the new solution is selected only if the sum of equality and inequality constraint violations is decreased. IPSO was tested with 24 benchmark problems and was compared with respect to state-of-the-art PSO approaches. The results obtained were competitive and even better in some cases, than those of the other algorithms with respect to which it was compared. The authors concluded that the proposed algorithm promoted a better exploration of the search space without adding any extra complexity to a traditional PSO algorithm.

The PSO-based approach reported in this paper maintains the simplicity of the original PSO algorithm, since it does not require gradient information (as in (Zahara and Hu 2008)), the use of external archives (as in (Muñoz-Zavala *et al.* 2005)) or specialized operators to perform a biased exploration of the search space (as in (Lu and Chen 2008)). Nevertheless, as it will be shown later on, the proposed approach provides competitive results with respect to state-of-the-art approaches that have been proposed for solving constrained optimization problems.

4. The Proposed Approach

The approach is called "Constrained Particle Swarm Optimization with a shake-mechanism" (CPSO-shake, for short). It inherits some characteristics from the classical PSO model (Eberhart and Kennedy 1995) and from a previous CPSO (Cagnina *et al.* 2006), but also incorporates features which aim to improve its performance, namely, a bi-population and a shake mechanism.

4.1. *Classical PSO Model*

The PSO algorithm operates on a population of individuals (the so-called *particles*). Such particles consist of vectors of real numbers, and each vector position is named *dimension*. The algorithm iterates searching for solutions and saves the best position found so far for the particles (for the "global best" or *gbest* model) or within the neighborhood (for the "local best" or *lbest* model). The best value reached by each particle (*pbest*) is also stored. The particles evolve using two update formulas, one for the particle's velocity and another for its position, in the following way:

$$v_{id} = \mathcal{X}(v_{id} + c_1 r_1 (p_{id} - par_{id}) + c_2 r_2 (M - par_{id})) \tag{5}$$

$$par_{id} = par_{id} + v_{id} \tag{6}$$

where v_{id} is the velocity of the particle i at the dimension d, $\mathcal{X}$ is the constriction factor (Clerc and Kennedy 2002) whose goal is to balance global exploration and local exploitation of the swarm, c_1 is the personal learning factor, and c_2 the social learning factor. r_1 and r_2 are two random numbers within the range [0, 1], which are used to introduce a stochastic value to determine how much of each factor is added. p_{id} is the dimension d of the best position reached by the particle i and M is either the best position reached by any particle in the neighborhood (pl_{id}) of the particle i or, the best position reached by any particle in the entire swarm (pg_d), depending of the model used (*lbest* or *gbest*). par_{id} is the value of the particle i at the dimension d.

4.2. *The Previous CPSO Approach*

Next, the modifications introduced in the PSO-based approach (Cagnina *et al.* 2006) with respect to the classical PSO model, are described.

4.2.1. *Updating Velocities and Particles*

A previous work (Cagnina *et al.* 2004) showed that it is possible to achieve quickly a reasonably good performance with the *gbest* model in a variety of problems. However, it is well known that the *gbest* model tends to lose diversity relatively quickly and, as a consequence of that, it tends to converge to local optima. The *lbest* model generally works better than the *gbest* model as a consequence of its higher capability to maintain diversity. Motivated by this, the approach proposed here has a formula to update the velocity which combines both the *gbest* (fast convergence) and the *lbest* (less susceptibility to being trapped in local optima) models. The idea is to replace equation (5) by equation (7) in the proposed CPSO.

$$v_{id} = \mathcal{X}(v_{id} + c_1 r_1 (p_{id} - par_{id}) + c_2 r_2 (pl_{id} - par_{id}) + c_3 r_3 (pg_d - par_{id})) \tag{7}$$

where c_3 is an additional social learning factor, r_3 is a random number within the range [0, 1], pl_{id} is the dimension d of the best position reached by any particle in the neighborhood of particle i, and pg_d is the dimension d of the best position reached by any particle in the swarm.
The equation for updating the particles is also modified. Instead of using equation (6) in all the iterations, it is selected with a probability of 0.925. The rest of the time an equation based on Kennedy's proposal (Kennedy 2003) is used, which is depicted in equation (8). In this case, the position of each particle is randomly chosen from a Gaussian distribution with the mean selected as the average between the best position recorded for the particle and the best in its neighborhood. The standard deviation is the difference between these two values.

$$par_i = N\left(\frac{p_i + pl}{2}, |p_i - pl|\right) \tag{8}$$

where par_i is the particle to be updated, N is the Gaussian random generator, p_i and pl are, respectively, the best position reached by the ith particle and, the best position reached by any particle in the neighborhood of par_i. The probability of selection adopted (0.925), deserves an explanation for being a seemingly unusual value. In order to reach this probability value, a Latin Hypercube Design (LHD) study, as suggested in the experimental design literature, was performed. LHD generates random points of the parameters to be set within a pre-defined range and fulfills the requirement suggested by specialists in experimental design, of being a space-filling approach. The parameter to be set was called probGauss, which is the complement of the probability of selection of interest (the probability of selection is 1-probGauss). In order to decide the range of

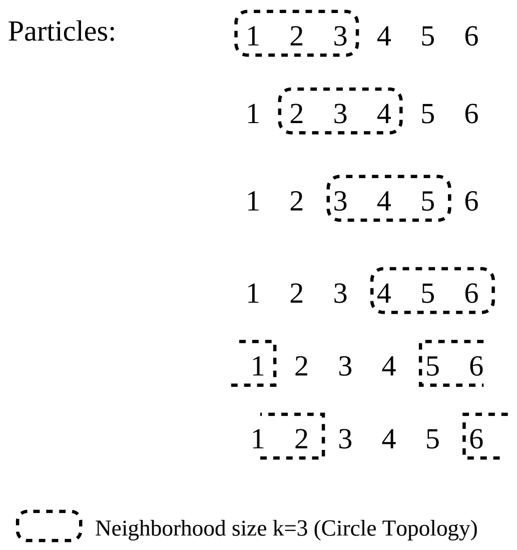

Figure 1. Possible neighborhoods.

interest for probGauss, another empirical study had to been done. The algorithm was tested using values for probGauss of 0.0, 0.1, 0.2, . . . , 0.9 (for each of these values, the test problems were evaluated). Positive results were found when using 0.0 and 0.1, which led to select [0.0, 1.0] as the pre-defined range for probGauss. Then, 20 LHD design points within this range were generated and, again, the same 19 test problems were evaluated. A number of 30 independent runs were performed with each test problems. The configuration having probGauss=0.075 provided the best overall results and was, therefore, chosen. Since probGauss=0.075, then the probability of selection was taken as 1-probGauss = 0.925.

4.2.2. *lbest Model: Circle Topology*

A number of different topologies have been proposed to implement the *lbest* model (Kennedy 1999). The present approach uses a simple topology which produced good results on the functions tested: the circle topology. In this model, each particle is connected to its neighbors, determining a neighborhood of size k, that is, there are $k - 1$ neighbors for each particle. The neighbors are determined by the (consecutive) position of the particles in the storage structure. Figure 1 illustrates this concept using a swarm of six particles and neighborhood size $k = 3$, so that each particle has two neighbors and six neighborhoods can be considered. For each neighborhood, the best particle (pl) needs to be determined and it is used in equations (7) and (8) for the particles within such neighborhood.

4.2.3. *Handling Constraints*

A variety of constraint-handling techniques have been proposed for evolutionary algorithms (Coello Coello 2002), but the one adopted in this work is quite simple. The constraint-handling method used in the proposed approach is based on the following rule: “a feasible particle is preferred over an infeasible one”. When the two particles compared are infeasible, the one closer to the feasible region is chosen. In order to do that, the algorithm stores the largest violation obtained for each constraint in each gen-

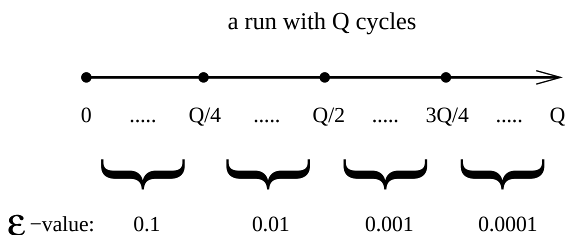

Figure 2. Variation of ϵ during a run of the PSO approach.

eration. When an individual is found to be infeasible, the sum of its constraints violations (this value is normalized with respect to the largest violation stored so far) is the one considered as its distance to the feasible region. This constraint-handling scheme is used when the *pbest*, *gbest* and *lbest* particles are chosen.

4.2.4. *Dynamic Tolerance*

As indicated before, in CPSO-shake, all the equality constraints are transformed into inequalities. The value of ϵ used is adapted three times during the entire run. When the search process starts, the value is initialized in 0.1. As the number of iterations is increased, the value of ϵ is divided by 10 at three different moments during the search (i.e., ϵ takes the values: 0.01, 0.001 and 0.0001 along a full execution of the algorithm). For example, assuming that Q is the total number of iterations to be performed, the value of ϵ will change according to the scheme graphically shown in Figure 2.

The main advantage of using a varying ϵ value is to favor the existence of feasible solutions at the beginning of the search process, by allowing almost-feasible solutions to be treated as feasible. In that way, the search space can be properly sampled (particularly in problems having equality constraints, which are normally hard to satisfy when using an evolutionary algorithm). As the number of iterations increases, ϵ is decreased, so that the approach starts converging towards solutions that satisfy the equality constraints with a higher accuracy.

It is worth indicating that the above adaptation procedure is applied exactly three times, to arrive to a final ϵ value of 0.0001. This value corresponds to the tolerance commonly adopted by other authors in their work (see for example Runarsson and Yao (2000)). Also, the experiments showed that applying the adaptation process three times was appropriate, since a lower number of times did not provide enough flexibility for the relaxation to have a significant impact on the search and a higher number did not allow sufficient time for the algorithm to reach the feasible region.

4.2.5. *Mutation Operator*

A dynamic mutation operator (Cagnina *et al.* 2004) was adopted for maintaining diversity into the swarms. The operator is applied to each individual with a certain probability (pm). Such probability is computed considering the total number of iterations performed by the algorithm ($cycles$) and the current iteration (cycle) number, using the following equation:

$$pm = max\_pm - \frac{max\_pm - min\_pm}{max\_cycle} * current\_cycle \tag{9}$$

where *max_pm* and *min_pm* are the maximum and minimum values that *pm* can take, *max_cycle* is the total number of cycles that the algorithm will iterate, and *current_cycle* is the current cycle in the iterative process. This operator is applied frequently at the beginning of the search process (exploration), and its application decays as the number of cycles increases (exploitation).

4.3. *CPSO-shake Model*

The modifications introduced to the previous CPSO approach are described next.

4.3.1. *Bi-population*

The idea of having several swarms in the population of a PSO algorithm has been adopted before by several researchers (Liang and Suganthan 2005, Blackwell and Branke 2006, Yen and Daneshyari 2006, Zhao *et al.* 2008, Trojanowski 2008). However, in most cases, there are several small and dynamic swarms which are frequently regrouped at a certain moment during the search, and in all cases the swarms exchange information. Here, this concept is used in a different way.

In the proposed CPSO-shake algorithm is applied the idea of maintaining more than one group of particles that explore the search space at the same time. The aim of this is to reduce the possibility of getting trapped in local optima. In the CPSO-shake algorithm, the entire population is split into only two static subpopulations, each of which is independently evolved. No information is shared between the two swarms. In the papers indicated before, the sizes of the swarms change over time because the particles move from one swarm to a different one. However, in the approach introduced here, the swarm sizes remain constant.

The issue that naturally arises here is why not to adopt more than two subpopulations. The pragmatic reason is: since the number of particles used in the population of the algorithm is small, it is considered that it would be inappropriate to adopt more than two populations. In fact, the neighborhood topology would not work properly if fewer particles were adopted in each sub-swarm than the current number and, therefore the reason for adopting only two subpopulations.

All the features stated before for the entire population (neighborhoods, *lbest* and *gbest* approaches, and equations for updating the velocity and the positions) remain without changes, but in this case, they are applied not to a single population, but to each subpopulation. When the iterative process finishes, the best particle from both subpopulations is reported as the final output.

It is worth noting that the scheme with two subpopulations indicated before could be parallelized using two or more processors in order to speed it up.

4.3.2. *Shake Mechanism*

In some previous related work (Cagnina *et al.* 2006), it was found that stagnation problems occur when trying to obtain values close to the optima for some difficult test functions. In order to overcome this problem, the algorithm reported in this paper incorporates a *shake mechanism.* This mechanism is applied when the percentage of infeasible individuals is higher than 10% (this value was empirically derived). It is worth noticing that it is not convenient to keep populations in which all the solutions are feasible, since infeasible solutions play an important role when trying to solve problems with active constraints, since they allow to explore the boundary between the feasible and infeasible regions.

In order to implement the shake mechanism, some particles are moved to a different place in the search space. Although this can be done by guiding a particle to a random direction, it is undesirable that the particles move away too much (the objective is to shake them just a little!). So, a particle with a good solution is selected as a reference: a randomly chosen *pbest* particle (pb_{SELd}). Thus, equation (10) is used to move a particle i:

$$v_{id} = \mathcal{X} v_{id} + c_1 r_1 (pb_{SELd}) \tag{10}$$

where v_{id} is the dth-position of the velocity vector, $\mathcal{X}$ is the constriction factor, c_1 is the personal learning factor multiplied by r_1, which is a random number within the range [0, 1]. pb_{SELd} is the dth-position of a (randomly chosen) selected *pbest* vector.

The shake mechanism is applied with a 50% probability over all the individuals in the swarm, at each iteration, and a different pb_{SELd} vector is chosen each time.

4.4. *CPSO-shake Pseudocode*

Figure 3 shows the pseudocode of the CPSO-shake algorithm. At the beginning of the search, the vectors of position and velocity of each particle in the entire population are initialized (lines 2 and 3). All particles are evaluated and the corresponding *pbest* values are saved (lines 4 and 5). In line 6, the *lbest* and *gbest* values are recorded (for the entire population) and when the swarm is divided into two different subpopulation, those values are the same for each sub-swarm (lines 6 and 7). In line 8, the ϵ value is initialized. Then, the subpopulations begin to evolve (line 9). During the evolutionary process, new values of *pbest*, *lbest* and *gbest* are chosen for each subpopulation and both, the velocity and the position of each particle, are updated (lines 10 to 25). At line 26, a *keeping* mechanism is applied in order to control that all the dimensions in all the particles are within the allowable bounds. When any of the dimensions falls outside its allowable bounds, this mechanism re-sets that dimension to its corresponding lower bound. Then, the shake mechanism is applied if the required conditions are fulfilled (lines 27 to 30). The mutation probability is updated and the particles are mutated, if applicable (lines 31 and 32). After that, the particles are evaluated, new "best" values are recorded, the ϵ value is updated and the percentage of infeasible particles is calculated (lines 33 to 35). All the process is repeated until the stop condition is reached. Finally, the best value reached by any subpopulation is taken and compared. The best of them is returned (lines 37 and 38).

5. Experimental Study

It is worth noting that although the approach seems to require many user-defined parameters (i.e., $\mathcal{X}$, c_1, c_2, c_3, *pmin*, *pmax*, ϵ, probability to select the equation to update the particles, size of the neighborhood, size of the population and number of cycles), this is not really the case. As will be seen next, many of these parameters can be set with constant values without the need to be tuned or empirically derived.

The relationship between the constriction factor $\mathcal{X}$ and the learning factors c_1, c_2 and c_3 has been studied by other researchers. Indeed, it has been proved (Kennedy and Eberhart 2001) that when the sum of learning factors exceeds 4.0, an explosion occurs,

```
0. CPSO-shake:
1. Swarm Initialization
2.   Initialize population
3.   Initialize velocities
4.   Evaluate fitness for each particle
5.   Record pbest for each particle
6.   Record lbest and gbest
7.   Split swarm in subpop1 and subpop2
8.   Initialize epsilon
9. Swarm flies through the search space
10.  DO
11.    FOR each subpop DO
12.        FOR i=1 TO numberOfparticles DO
13.            Search the best leader in the
14.               neighborhood of $part_i$
15.               and store it in $lbest_i$
16.            FOR j=1 TO numberOfdimensions DO
17.                Update $vel_{ij}$
18.                IF probability>$(0.075)$
19.                   Update $part_{ij}$ with eq.(6)
20.                ELSE
21.                   Gaussian update with eq.(8)
22.                END
23.            END
24.        END
25.    END
26.    Keep particles
27.    Test for shake-mechanism
28.    IF % infeasibles > 10%
29.       shake-mechanism
30.    END
31.    Update $pm$
32.    Mutate every particle depending on $pm$
33.    Evaluate fitness($part_i$)
34.    Record pbest and gbest
35.    Update epsilon
36.  WHILE($current\_cycle < max\_cycle$)
37.  result=BEST($best\_subpop1$,$best\_subpop2$)
38.  RETURN(result)
```

Figure 3. Pseudocode of CPSO-shake.

which keeps PSO from converging to the global optimum. In order to avoid that, the constriction factor $\mathcal{X}$ was adopted to regulate this effect in order to maintain a proper behavior of the PSO algorithm during the search (Clerc and Kennedy 2002). For the purposes of the work reported here, it was decided to set only one of the learning factors and set the other two to the same value, because there is equal preference for each of the two factors (social or individual). In other words, a value in the range [1.4, 1.9] was adopted for the three learning factors. The constriction factor was set as $\mathcal{X} = c_1 - 1.0$ in all cases, so it can vary in the range [0.4, 0.9] which follows the recommendations from (Clerc and Kennedy 2002). This reduces the setting of four parameters to that of setting only one.

The maximum ϵ value was set to 0.0001, as discussed in Section 4.2.4. The probability to select the equation (normal or Gaussian) to update the particles was chosen so that it favored the use of the first, as discussed in Section 4.2.1.

The size of neighborhood was fixed to 3 for all the examples adopted here, because it was empirically found that larger neighborhoods combined with a low number of particles in each swarm (less than 20) leads to a poor performance.

The performance of the proposed approach was assessed using 24 (nonlinear) constrained optimization problems that have been proposed in the specialized literature. Additionally, the proposed approach was also applied to three truss optimization problems. The test problems and the corresponding experimental setup adopted are described next.

5.1. *A Benchmark of Constrained Optimization Problems*

The 24 test problems adopted here were reported in (Liang *et al.* 2006), from which the first 13 correspond to the test problems adopted in (Runarsson and Yao 2000) and in many other publications related to evolutionary constrained optimization (see for example (Toscano-Pulido and Coello Coello 2004, Muñoz-Zavala *et al.* 2005, Zahara and Hu 2008)). The detailed descriptions of these test problems are available in (Liang *et al.* 2006). It is worth noticing that all of these test functions were transformed into minimization problems. Also, all the equality constraints were transformed into inequalities using a tolerance ϵ, as indicated before (see Section 4.2.4). For each problem, 25 independent runs were performed with a total of 350,000 objective function evaluations per run. The proposed CPSO-shake used the following parameters: swarm size = 10 particles, $pm\_min = 0.1$, $pm\_max = 0.4$, $c_1 = 1.8$. These parameters were empirically derived after performing an analysis of variance (ANOVA) of the main parameters of the approach. This ANOVA allowed to identify the combination of parameter values that provided the best overall performance for the proposed approach.

Results were compared with respect to those obtained by DMS-C-PSO (Liang and Suganthan 2006) and IPSO (Mezura-Montes and Flores-Mendoza 2009), which are two PSO-based algorithms that have been found to be very competitive in the benchmark adopted here (see (Liang and Suganthan 2006, Mezura-Montes and Flores-Mendoza 2009)). The parameters setting for DMS-C-PSO was as suggested in (Liang and Suganthan 2006), except for the number of fitness function evaluations,[1] which was set at 350,000 in order to allow a fair comparison with respect to CPSO-shake. The parameters setting for IPSO was as suggested in (Mezura-Montes and Flores-Mendoza 2009) except for the number of iterations and the number of independent runs used to obtain the best results. In (Mezura-Montes and Flores-Mendoza 2009), the authors performed 30 independent runs of IPSO, each of which consumed 2,000 generations, for a total of 160,000 fitness function evaluations per run. In order to allow a fair comparison with CPSO-shake, a new

[1]In (Liang and Suganthan 2006), the authors adopted 300,000, 400,000 and 500,000 fitness function evaluations.

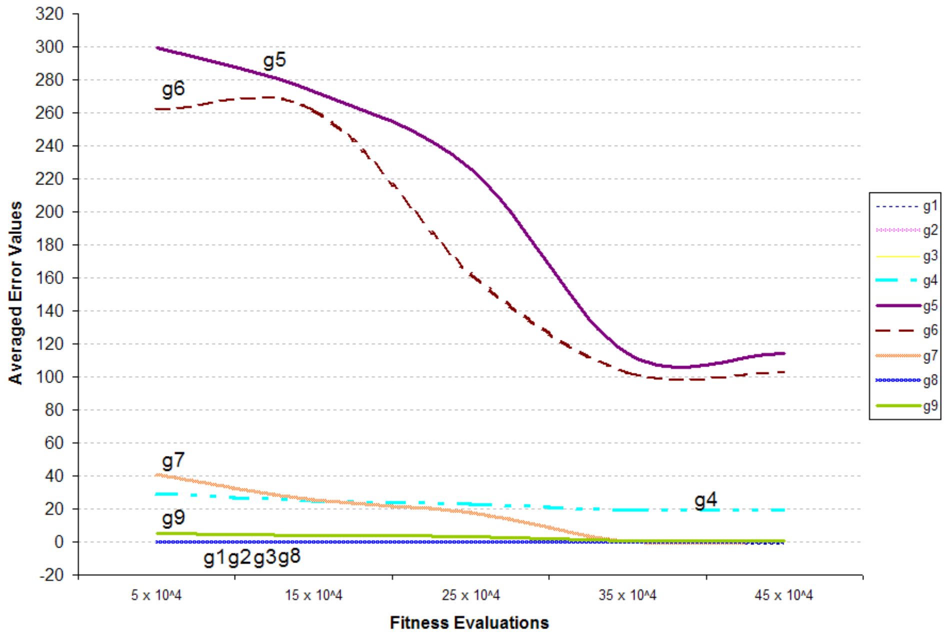

Figure 4. Evolution curves for functions g1-g9.

experimental study was conducted, in which IPSO was run for 4,375 generations in order to reach a total of 350,000 fitness function evaluations. In this case, 25 independent runs were performed. The other parameter values were set as in (Mezura-Montes and Flores-Mendoza 2009), since the authors of IPSO reported that such parameter values provided the most consistent performance of their approach. In Figures 4, 5 and 6, the evolution curves of CPSO-shake for the test functions from 1 to 9, from 10 to 19 and from 20 to 24, respectively, are presented. The vertical axis shows the distance between the best values reached (the average) and the global optimum (or best known value), and the horizontal axis shows the number of fitness function evaluations (FEs) performed over the 25 runs. As can be observed in Figure 4, the average error values for g1, g2, g3, g8 and g9 are zero which indicates that the algorithm found the optimum (or very close values) with 50,000, 150,000, 250,000 and 350,000 FEs, respectively. The same happens in Figure 5 with functions g11, g12, g13, g15, g16 and g18. For functions g4 and g19 the best values obtained are slightly improved when the number of FEs is increased. However, a significant improvement is shown in the rest of the functions: g5, g6, g7, g10, g14 and g17. In Figure 6, it can be observed that for g20 and g24 the average error is zero. For g21 and g22 the best values found improve as the number of iterations increases and for g23 a significant improvement is observed.

The best values reached by each algorithm are shown in Table 1. It is possible to observe that, in general, CPSO-shake was able to obtain the best known values (g1, g2, g3, g8, g9, g11, g12, g15, g16, g17, g18 and g24). For functions g4, g5, g6, g7, g10 and g14, CPSO-shake found solutions very close to the optimum. The best solutions found are feasible in all cases except the instances indicated with (*) in Table 1. For g20, g21 and g22, CPSO-shake did not find any feasible solutions. CPSO-shake outperformed DMS-C-PSO in functions g1, g3, g4, g5, g6, g7, g9, g10, g13, g16, g17, g18 and g19.

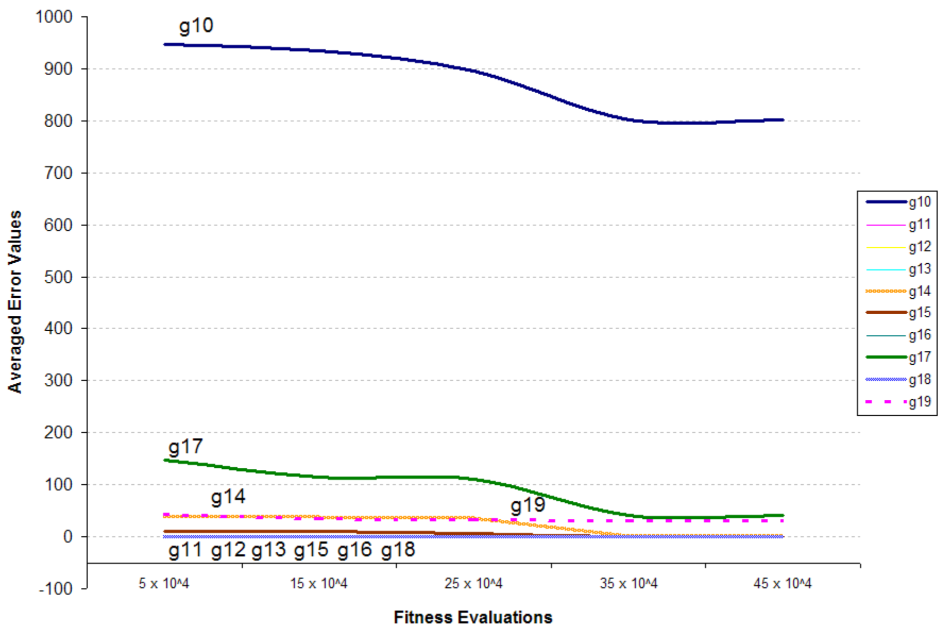

Figure 5. Evolution curves for functions g10-g19.

For g20, g21, g22 and g23 DMS-C-PSO did not find any feasible solutions. CPSO-shake outperformed IPSO in 9 functions (g2, g3, g5, g7, g10, g13, g14, g17 and g23), from which, in 6 cases the differences were significant (g3, g7, g10, g13, g17 and g23). In contrast, IPSO outperformed CPSO-shake in only 4 functions (g4, g6, g19 and g21) and from them, only in the last one the difference was significant. For g20 and g22, IPSO did not find any feasible solutions. The comparisons with DMS-C-PSO and IPSO were direct, since that the source codes of both algorithms were obtained from their authors.

The deviations of the mean values obtained by the algorithms, with respect to their reference solutions are presented in Table 2. ¿From Table 2 it is possible to observe that the proposed CPSO-shake algorithm presents some variability in the results that it obtained. It is worth noting, however, that the mean errors of DMS-C-PSO are, in general, worse than those of CPSO-shake. However, the mean errors of IPSO are, in general, better than those obtained by CPSO-shake, although, in several cases the differences are negligible. ¿From Table 2, it should be clear that the robustness of the proposed approach is an aspect that still needs to be improved.

The overall conclusion from this first comparative study is that the approach is competitive with respect to other competitive PSO-based approaches.

5.2. *Engineering Optimization Problems*

The second comparative study performed, adopted truss optimization problems that have been widely studied in the specialized literature (Landa Becerra and Coello Coello 2006, Bernardino *et al.* 2007). For this study, 3 truss optimization problems taken from (Belegundu 1982) were adopted: a 10-bar plane truss (modeled with 20 design variables corresponding to the dimension (height and width) of each element), a 25-bar space

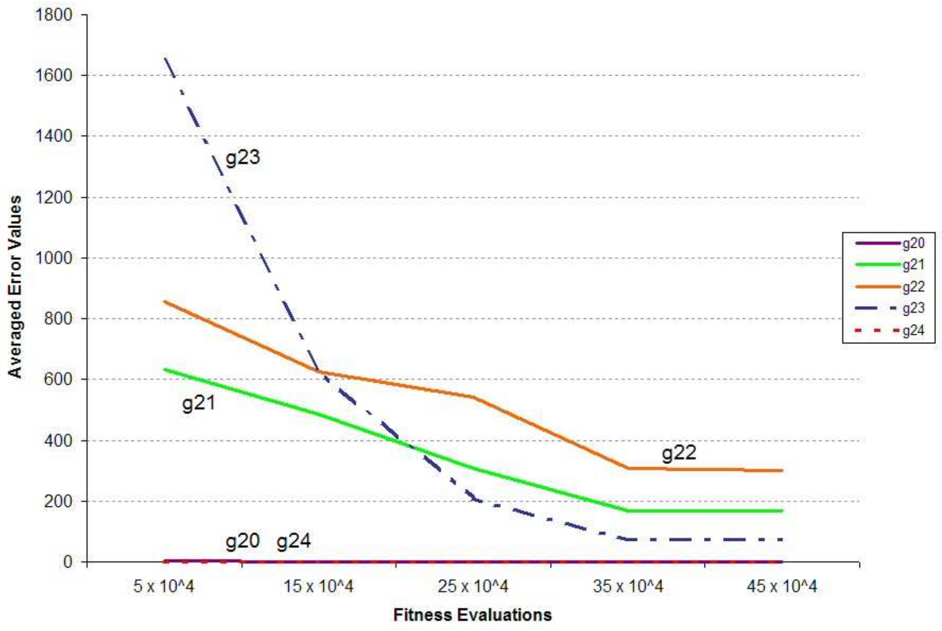

Figure 6. Evolution curves for functions g20-g24.

Table 1. Comparison of the best values obtained by the proposed CPSO-shake, DMS-C-PSO and IPSO after performing 350,000 objective function evaluations. The column labeled with BKV reports the optimum or best known value for that specific test problem.

Test problem	BKV	CPSO-shake	DMS-C-PSO	IPSO
g1	-15.000	**-15.000**	-14.770	**-15.000**
g2	-0.803	**-0.803**	**-0.803**	-0.792
g3	-1.000	**-1.000**	-0.998	-0.614
g4	-30,665.539	-30,665.538	$-32,217.121^*$	**-30,665.539**
g5	5,126.496	5,126.498	5,132.459	5,126.546
g6	-6,961.813	-6,961.825	-6,961.910	-6,961.814
g7	24.306	24.309	24.741	24.339
g8	-0.095	**-0.095**	**-0.095**	**-0.095**
g9	680.630	**680.630**	680.635	**680.630**
g10	7,049.248	7,049.285	7,498,872	7,084.351
g11	0.749	**0.749**	**0.749**	**0.749**
g12	-1.000	**-1.000**	**-1.000**	**-1.000**
g13	0.053	0.054	0.072	0.084
g14	-47.764	-47.635	-47.635	-47.631
g15	961.715	**961.715**	**961.715**	**961.715**
g16	-1.905	**-1.905**	-2.579^*	**-1.905**
g17	8,853.539	**8,853.539**	8,856.526	8,858.361
g18	-0.866	**-0.866**	-0.865	**-0.866**
g19	32.655	34.018	34.730	33.537
g20	0.097^*	0.256^*	0.664^*	8.170^*
g21	193.724	361.846^*	253.590^*	193.739
g22	236.430	545.112^*	0.000^*	$2,248.269^*$
g23	-400.055	-326.963	$-2,294.500^*$	-259.229
g24	-5.508	**-5.508**	**-5.508**	**-5.508**

*Infeasible solution.

Table 2. Errors of the mean values obtained by each algorithm, with respect to the optimum or best known values.

Test problem	CPSO-shake	DMS-C-PSO	IPSO
g1	0	1.822	0.08
g2	0.007	0.040	0.106
g3	0	0.011	0.803
g4	19.360	1,552.439	0
g5	114	263.296	20.271
g6	102.738	554.023	0.001
g7	0.606	1.576	0.338
g8	0	0.001	0
g9	0.743	0.085	0.022
g10	801.153	2,727.872	350.201
g11	0	0	0
g12	0	0	0
g13	0.397	0.517	0.381
g14	2.099	0.130	2.468
g15	0.801	2.226	0.587
g16	0.110	0.011	0
g17	41.169	70.461	48.454
g18	0.079	0.002	0.004
g19	31.850	2.074	3.590
g20	2.679	5.368	15.252
g21	23.673	104.905	30.371
g22	56.334	1,215.442	17,565.760
g23	128.212	398.948	314.394
g24	0	0	0

truss and a 200-bar plane truss. The detailed descriptions of these test problems may be consulted in the Appendix at the end of this paper.

The proposed CPSO-shake used the following parameter settings: swarm size = 10 particles and $c_1 = 1.7$. The probability of mutation was set to zero, that is, no mutation operator was applied. Those settings were empirically derived after numerous experiments.

30 independent runs were performed per problem, with a total of 100,100 objective function evaluations per run. This number of evaluations was chosen because it is the lowest value reported in the previous works adopted here for the comparative study reported (see (Landa Becerra and Coello Coello 2006), in which the use of a cultured differential evolution approach is used).

The results of CPSO-shake were compared with respect to the following approaches:

(1) Several mathematical programming techniques adopted in (Belegundu 1982): Feasible directions (CONMIN and OPTDYN), Pshenichny's Recursive Quadratic Programming (LINRM), Gradient Projection (GRP-UI), and Exterior Penalty Function (SUMT).
(2) Two simple genetic algorithms with a penalty function: SGA (Galante 1992) and GENETICO (Coello Coello *et al.* 1994).
(3) A gradient-based method for constrained optimization, embedded in a computer program called BEHSAZ (BEHP) (Memari and Fuladgar 1994).
(4) Nonlinear goal programming (NLP) (El-Sayed and Jang 1994).
(5) Harmony search (HSA) (Lee and Geem 2004).
(6) A genetic algorithm with an adaptive penalty scheme (AP-GA) (Lemonge and

Table 3. Solution vector for the 10-bar plane truss.

	Value
x_0	104.714546
x_1	88.240891
x_2	0.400003
x_3	0.400000
x_4	120.564369
x_5	46.732246
x_6	98.870956
x_7	19.894732
x_8	0.400006
x_9	0.400001
x_{10}	3.411510
x_{11}	0.400211
x_{12}	63.583042
x_{13}	4.455610
x_{14}	42.467812
x_{15}	74.717102
x_{16}	136.491806
x_{17}	30.628040
x_{18}	0.400001
x_{19}	0.400000
Displacement	7.591675
Stress	84.707460
Weight	4,656.361619

Barbosa 2004).

(7) A modified version of the Pareto Archived Evolution Strategy (PAES) (Knowles and Corne 2000), which is used for single-objective constrained optimization (IS-PAES) (Hernández-Aguirre *et al.* 2004).

(8) Differential evolution (Price 1999) hybridized with a cultural algorithm (CDE) (Landa Becerra and Coello Coello 2006).

(9) A particle swarm optimization approach for structural design (IPSO) (Perez and Behdinan 2007a).

(10) A heuristic particle swarm optimizer (HPSO) (Li *et al.* 2007).

(11) A hybrid of a genetic algorithm and an artificial immune system (AIS-GA) (Bernardino *et al.* 2007).

In Tables 3, 4 and 5, the decision variables corresponding to the best solutions found by CPSO-shake are shown.

In Table 6, the comparison of results (objective function values) found by CPSO-shake and those obtained by the previously indicated algorithms is shown, for the three truss optimization problems adopted. Note that the best value reported by (Landa Becerra and Coello Coello 2006) for the 200-bar plane truss is not included, because they used different input data for that problem.

Table 7 summarizes the best objective function values (**Best Known**) reported for each of the three trusses adopted (considering all the algorithms previously indicated and CPSO-shake). The worst (**Worst**) values obtained and the corresponding standard deviations (**Std.Dev**) are also shown. It can be clearly seen that CPSO-shake obtained better results than the best known for each of the three trusses. Additionally, its standard deviations are lower than those obtained with the other approaches, except for the 10-bar

Table 4. Solution vector for the 25-bar plane truss.

	Value
x_0	0.100007
x_1	0.100054
x_2	3.592288
x_3	0.100000
x_4	1.978739
x_5	0.777161
x_6	0.148297
x_7	3.926824
Displacement	1.559454
Stress	91,679.728643
Weight	467.307565

Table 5. Solution vector for the 200-bar plane truss.

	Value		Value
x_0	0.099778	x_{16}	0.059441
x_1	0.827075	x_{17}	50.606869
x_2	0.010058	x_{18}	0.010290
x_3	0.010004	x_{19}	66.012398
x_4	3.645809	x_{20}	0.279517
x_5	0.049748	x_{21}	0.746613
x_6	0.010007	x_{22}	94.119659
x_7	10.451566	x_{23}	0.488295
x_8	0.010109	x_{24}	114.492180
x_9	16.537201	x_{25}	1.489903
x_{10}	0.090882	x_{26}	24.173244
x_{11}	0.057575	x_{27}	69.310654
x_{12}	23.390146	x_{28}	168.946198
x_{13}	0.175855	Displacement	45.352070
x_{14}	34.062832	Stress	1,105.103661
x_{15}	0.236381	Weight	22,705.327292

plane truss. This seems to contradict the results obtained with the benchmark problems. However, possibly the reason for this apparent increase in robustness is related to the fact that the evolutionary algorithms that tend to produce the best final results in engineering optimization problems are normally more explorative, which makes them generate poorer solutions from time to time. The price that must be paid for this higher explorative power is precisely a higher standard deviation in the results, as can be seen here. The improvements achieved in the truss optimization problems are not the same in all cases: for the 10-bar plane truss the improvement is negligible, for the 25-bar space truss, there is a 4% improvement, but in the 200-bar plane truss, the improvement almost reaches 18% with respect to the reference solution. These results seem to indicate that the proposed approach is a viable alternative for solving engineering optimization problems.

Table 6. Comparison of the **best** values obtained by CPSO-shake and other algorithms. The '-' indicates that the authors did not report that value.

Algorithm	10-bar-truss	25-bar-truss	200-bar-truss
CONMIN (Belegundu 1982)	4,793.00	-	34,800.00
OPTDYN (Belegundu 1982)	9,436.00	-	-
LINRM (Belegundu 1982)	6,151.00	-	33,315.00
GRP-UI (Belegundu 1982)	5,077.00	-	-
SUMT (Belegundu 1982)	5,070.00	-	27,564.00
SGA (Galante 1992)	4,987.00	-	-
BEHP (Memari and Fuladgar 1994)	4,981.10	-	-
GENETICO (Coello Coello *et al.* 1994)	5,691.82	539.48	-
NLP (El-Sayed and Jang 1994)	5,013.24	-	-
HSA (Lee and Geem 2004)	5,057.88	544.38	-
AP-GA (Lemonge and Barbosa 2004)	5,069.09	-	-
ISPAES (Hernández-Aguirre *et al.* 2004)	5,951.00	569.80	-
CDE (Landa Becerra and Coello Coello 2006)	4,656.39	-	-
IPSO (Perez and Behdinan 2007a)	5,024.21	485.33	-
HPSO (Li *et al.* 2007)	5,060.92	545.19	-
AIS-GA (Bernardino *et al.* 2007)	5,062.67	-	-
CPSO-shake	**4,656.36**	**467.30**	**22,705.32**

Table 7. Comparison of the **best** values obtained by CPSO-shake and the best values reported by other approaches. The '-' indicates that the authors did not report that value.

	Other Approaches			CPSO-shake		
Truss	Best Known	Std.Dev	Worst	Best Found	Std.Dev	Worst
10-bar	4,656.39	0.18	4,656.71	**4,656.36**	2.84	4,696.06
25-bar	485.33	-	534.84	**467.30**	0.35	470.87
200-bar	27,564.00	-	-	**22,705.32**	1,566.65	30,107.62

6. Conclusions and Future Work

This paper has introduced a PSO-based approach for constrained optimization, called CPSO-shake. This approach introduces relatively simple changes to a traditional PSO algorithm, aiming to provide better diversity maintenance and better exploration of constrained search spaces. The approach was validated using both traditional test problems adopted in the evolutionary optimization literature and engineering optimization problems. Results were compared with respect to other approaches, including a variety of evolutionary algorithms as well as mathematical programming techniques. In both cases, the results were found to be very encouraging, and place CPSO-shake as a highly competitive PSO-based optimizer for nonlinear constrained problems.

It would be interesting to work in the future in the design of mechanisms that can improve the robustness of the proposed approach (i.e., that can reduce its variability of results over several independent runs). In that regard, possibly the use of local search could be quite useful. Additionally, it would be interesting to experiment with other PSO variants and with other constraint-handling mechanisms, including a possible relaxation of the inequality constraints (instead of only relaxing the equality constraints). We would also like to test our proposed approach with other benchmarks (see for example (Mallipeddi and Suganthan 2010a)) and to compare our results with other PSO-based approaches that have been recently introduced (see for example (Liang *et al.* 2010)). The development of a parallel version of the proposed approach would also be interesting, since it would allow its use in problems having computationally expensive objective functions. Such a parallel implementation could also contain different PSO-based approaches (one in each processor) in order to allow the search skills of different PSO-based variants to be combined (throught the use of migration). This is similar to the idea of an "ensemble" of constraint-handling techniques recently introduced in (Mallipeddi and Suganthan 2010b).

Acknowledgments

The authors thank P. N. Suganthan for providing the source code of DMS-C-PSO as well as Efrén Mezura-Montes and Jorge Isacc Flores-Mendoza for providing the source code of IPSO, both of which were used in the experiments reported here.

The first and second author gratefully acknowledge the continuous support from ANPCyT and the Universidad Nacional de San Luis. The third author acknowledges support from CONACyT project no. 103570.

References

Belegundu, A.D., 1982. A Study of Mathematical Programming Methods for Structural Optimization. Thesis (PhD). University of Iowa, Iowa, USA.

Bernardino, H., Barbosa, H., and Lemong, A., 2007. A Hybrid Genetic Algorithm for Constrained Optimization Problems in Mechanical Engineering. *In*: *2007 IEEE Congress on Evolutionary Computation (CEC 2007).* Singapore, 25-28 September. IEEE Press, 646–653.

Blackwell, T. and Branke, J., 2006. Multiswarms, exclusion, and anti-convergence in dynamic environments. *IEEE Transactions on Evolutionary Computation*, 10 (4), 459–472.

Bochenek, B. and Forys, P., 2006. Structural optimization for post-buckling behavior using particle swarms. *Structural and Multidisciplinary Optimization*, 32 (6), 521–531.

Cagnina, L., Esquivel, S., and Gallard, R., 2004. Particle Swarm Optimization for sequencing problems: a case study. *In*: *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC 2004).* Portland, Oregon, USA. 20-23 June. IEEE Press, 536–541.

Cagnina, L.C., Esquivel, S.C., and Coello Coello, C.A., 2006. A Particle Swarm Optimizer for Constrained Numerical Optimization. *In*: T.P. Runarsson, H.G. Beyer, E. Burke, J.J. Merelo-Guervós, L.D. Whitley and X. Yao, eds. *Parallel Problem Solving from Nature (PPSN IX). 9th International Conference.* Reykjavik, Iceland. 9-13 September. Lecture Notes in Computer Science Vol. 4193. Springer, 910–919.

Chootinan, P. and Chen, A., 2006. Constraint Handling In Genetic Algorithms Using A Gradient-Based Repair Method. *Computers and Operations Reseach*, 33 (8), 2263–2281.

Clerc, M. and Kennedy, J., 2002. The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, 6 (1), 58–73.

Coello Coello, C.A., 2002. Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191 (11-12), 1245–1287.

Coello Coello, C.A., Rudnick, M., and Christiansen, A.D., 1994. Using Genetic Algorithms for Optimal Design of Trusses. *In*: *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence.* New Orleans, Louisiana, USA. 6-9 November. IEEE Computer Society Press, 88–94.

Eberhart, R. and Kennedy, J., 1995. A new optimizer using particle swarm theory. *In*: *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS'95.* Nagoya, Japan. 4-6 October. IEEE Press, 39–43.

Eberhart, R.C. and Shi, Y., 2000. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. *In*: *Proceedings of the 2000 IEEE Congress on Evolutionary Computation (CEC'2000).* Piscataway, New Jersey, USA. 16-19 July IEEE Press, 84–88.

El-Sayed, M. and Jang, T., 1994. Structural optimization using unconstrained non-linear goal programming algorithm. *Computers and Structures*, 52 (4), 723–727.

Galante, M., 1992. Structures Optimization by a simple genetic algorithm. *In*: *Numerical methods in engineering and applied sciences.* Barcelona, Spain: CIMNE, 862–870.

Gere, J.M. and Weaver, W., 1965. *Analysis of Framed Structures.* D. Van Nostrand Company, Inc.

Hamida, S.B. and Schoenauer, M., 2002. ASCHEA: New Results Using Adaptive Segregational Constraint Handling. *In*: *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002).* Honolulu, Hawaii. 12-17 May. IEEE Service Center, 884–889.

He, Q. and Wang, L., 2007. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, 20 (1), 89–99.

Hedar, A. and Fukushima, M., 2006. Derivative-free filter simulated annealing method for constrained continuous global optimization. *Journal of Global Optimization*, 35 (4), 521–549.

Hernández-Aguirre, A., *et al.*, 2004. Handling Constraints using Multiobjective Optimization Concepts. *International Journal for Numerical Methods in Engineering*, 59 (15), 1989–2017.

Hu, X. and Eberhart, R., 2002. Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization. *In*: *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, Vol. 5. Orlando, USA. 14-18 July. IIIS.

Hu, X., Eberhart, R.C., and Shi, Y., 2003. Engineering Optimization with Particle Swarm. *In*: *Proceedings of the 2003 IEEE Swarm Intelligence Symposium.* Indianapolis, Indiana, USA. 24-26 April. IEEE Service Center, 53–57.

Kennedy, J., 1999. Small World and Mega-Minds: effects of neighborhood topologies on Particle Swarm Performance. *In*: *Proceedings of the 1999 IEEE Congress on Evolutionary Computation (CEC 1999).* Washington, USA. 6-9 July IEEE Service Center, 1931–1938.

Kennedy, J., 2003. Bare Bones Particle Swarms. *In*: *Proceedings of the IEEE 2003 Swarm Intelligence Symposium (SIS 2003).* Indianapolis, Indiana, USA. 24-26 April. IEEE Press, 80–87.

Kennedy, J. and Eberhart, R.C., 1999. The particle swarm: social adaptation in information processing systems. *In*: D. Corne, M. Dorigo and F. Glover, eds. *New Ideas* *in Optimization.* London, UK: McGraw-Hill, 379–388.

Kennedy, J. and Eberhart, R.C., 2001. *Swarm Intelligence.* California, USA: Morgan Kaufmann Publishers.

Knowles, J.D. and Corne, D.W., 2000. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8 (2), 149–172.

Koziel, S. and Michalewicz, Z., 1999. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7 (1), 19–44.

Landa Becerra, R. and Coello Coello, C.A., 2006. Cultured differential evolution for constrained optimization. *Computer Methods in Applied Mechanics and Engineering*, 195 (33–36), 4303–4322.

Lee, K.S. and Geem, Z.W., 2004. A new structural optimization method based on the harmony search algorithm. *Computers and Structures*, 82 (9–10), 781–798.

Lemonge, A. and Barbosa, H., 2004. An Adaptive Penalty Scheme for Genetic Algorithms in Structural Optimization. *International Journal for Numerical Methods in Engineering*, 59 (5), 703–736.

Li, L.J., *et al.*, 2007. A heuristic particle swarm optimizer for optimization of pin connected structures. *Computers and Structures*, 85 (7–8), 340–349.

Liang, J.J., *et al.*, *Problem Definitions and Evaluation Criteria for the CEC 2006*, Nanyang Technological University, Singapore. 2006.

Liang, J.J. and Suganthan, P.N., 2005. Dynamic multi-swarm particle swarm optimizer. *In*: *Proceedings of the IEEE 2005 Swarm Intelligence Symposium (SIS 2005).* California, USA. 8-10 June. IEEE Press, 124–129.

Liang, J.J. and Suganthan, P.N., 2006. Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constrain-Handling Mechanism. *In*: *2006 IEEE Congress on Evolutionary Computation (CEC 2006).* Vancouver, BC, Canada. 16-21 July. IEEE, 316–323.

Liang, J., Zhigang, S., and Zhihui, L., 2010. Coevolutionary Comprehensive Learning Particle Swarm Optimizer. *In*: *2010 IEEE Congress on Evolutionary Computation (CEC'2010)*, July 18–23. Barcelona, Spain: IEEE Press, 1505–1512.

Lu, H. and Chen, W., 2006. Dynamic-objective particle swarm optimization for constrained optimization problems. *Journal of Combinatorial Optimization*, 12 (4), 409–419.

Lu, H. and Chen, W., 2008. Self-adaptive velocity particle swarm optimization for solving constrained optimization problems. *Journal of Global Optimization*, 41 (3), 427–445.

Mallipeddi, R. and Suganthan, P.N., *Problem Definitions and Evaluation Criteria for the CEC 2010 Competition on Constrained Real-Parameter Optimization*, Technical report, Nanyang Technological University, Singapore. 2010a.

Mallipeddi, R. and Suganthan, P.N., 2010b. Ensemble of Constraint Handling Techniques. *IEEE Transactions on Evolutionary Computation*, 14 (4), 561–579.

Memari, A. and Fuladgar, A., 1994. Minimum Weight Design of Trusses by BEHSAZ Program. *In*: *Proceedings of the 2nd International Conference on Computational Structures Technology.* Athens, Greece. August 30–September 1. Civil-Comp Press, 179–185.

Mezura-Montes, E. and Flores-Mendoza, J.I., 2009. Improved Particle Swarm Optimization in Constrained Numerical Search Spaces. *In*: R. Chiong, ed. *Nature-Inspired Algorithms for Optimisation - Nature-Inspired Algorithms for Optimisation.* Springer, 299–332.

Mezura-Montes, E., ed. , 2009. *Constraint-Handling in Evolutionary Optimization.*

Berlin/Heidelberg: Springer. ISBN 978-3-642-00618-0.

Mezura-Montes, E. and López-Ramírez, B.C., 2007. Comparing Bio-Inspired Algorithms in Constrained Optimization Problems. *In*: *2007 IEEE Congress on Evolutionary Computation (CEC 2007).* Singapore, 25-28 September. IEEE Press, 662–669.

Muñoz-Zavala, A.E., Aguirre, A.H., and Diharce, E.R.V., 2005. Constrained Optimization via Particle Evolutionary Swarm Optimization Algorithm (PESO). *In*: H.G.B. et al., ed. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2005)*, Vol. 1, June. ISBN 1-59593-010-8 New York: ACM Press, 209–216.

Muñoz-Zavala, A.E., *et al.*, 2006. PESO+ for Constrained Optimization. *In*: *2006 IEEE Congress on Evolutionary Computation (CEC 2006).* Vancouver, BC, Canada. 16-21 July. IEEE, 935–942.

Nelder, J.A. and Mead, R., 1965. A Simplex Method for Function Minimization. *The Computer Journal*, 7, 308–313.

Paquet, U. and Engelbrecht, A., 2003. A New Particle Swarm Optimiser for Linearly Constrained Optimization. *In*: *Proceedings of the Congress on Evolutionary Computation 2003 (CEC 2003).* Canberra, Australia. 8-12 December. IEEE Service Center, 227–233.

Parsopoulos, K. and Vrahatis, M., 2005. Unified Particle Swarm Optimization for solving constrained engineering optimization problems. *Advances in Natural Computation, Pt. 3. Lecture Notes in Computer Science Vol. 3612*, 582–591.

Perez, R.E. and Behdinan, K., 2007a. Particle Swarm approach for Structural Design Optimization. *Computers and Structures*, 85 (19-20), 1579–1588.

Perez, R.E. and Behdinan, K., 2007b. Particle Swarm Optimization in Structural Design. *In*: F.T. Chan and M.K. Tiwari, eds. *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization.* ISBN 978-3-902613-09-7 Vienna, Austria: Itech Education and Publishing, 373–394.

Price, K.V., 1999. An Introduction to Differential Evolution. *In*: D. Corne, M. Dorigo and F. Glover, eds. *New Ideas in Optimization.* London, UK: McGraw-Hill, 79–108.

Rajeev, S. and Krishnamoorthy, C.S., 1997. Genetic Algorithms-based methodologies for design optimization of trusses. *Journal of Structural Engineering*, 123 (3), 350–358.

Runarsson, T.P. and Yao, X., 2000. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4 (3), 284–294.

Runarsson, T.P. and Yao, X., 2005. Search biases in constrained evolutionary optimization. *IEEE Transactions on Systems, Man, and Cybernetics Part C—Applications and Reviews*, 35 (2), 233–243.

Toscano-Pulido, G. and Coello Coello, C.A., 2004. A Constraint-Handling Mechanism for Particle Swarm Optimization. *In*: *Proceedings of the Congress on Evolutionary Computation 2004 (CEC 2004).* Portland, Oregon, USA. 20-23 June. Piscataway, New Jersey: IEEE Service Center, 1396–1403.

Trojanowski, K., 2008. Multi-Swarm That Learns. *In*: *Intelligent Information Systems 2008* Vancouver, BC, Canada: IEEE, 121–130.

Ye, D., Chen, Z., and Liao, J., 2007. A New Algorithm for Minimum Attribute Reduction Based on Binary Particle Swarm Optimization with Vaccination. *In*: Z.H. Zhou, H. Li and Q. Yang, eds. *Advances in Knowledge Discovery and Data Mining, 11th Pacific-Asia Conference, PAKDD 2007.* Nanjing, China. 22-25 May. Springer. Lecture Notes in Computer Science Vol. 4426, 1029–1036.

Yen, G.G. and Daneshyari, D., 2006. Diversity-based Information Exchange among Multiple Swarms in Particle Swarm Optimization. *In*: *2006 IEEE Congress on Evolu-*

tionary Computation (CEC 2006), Vancouver, BC, Canada. 16-21 July. IEEE Press, 1686–1693.

Zahara, E. and Hu, C.H., 2008. Solving constrained optimization problems with hybrid particle swarm optimization. *Engineering Optimization*, 40 (11), 1031–1049.

Zhang, W.J. and Xie, X.F., 2003. DEPSO: Hybrid Particle Swarm with Differential Evolution Operator. *In*: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC 2003)*, Vol. 4. Washington DC, USA. 5-8 October. IEEE, 3816–3821.

Zhang, W.J., Xie, X.F., and Bi, D.C., 2004. Handling Boundary Constraints for Numerical Optimization by Particle Swarm Flying in Periodic Search Space. *In*: *Proceedings of the Congress on Evolutionary Computation 2004 (CEC 2004).* Portland, Oregon, USA. 20-23 June. Piscataway, New Jersey: IEEE Service Center, 2307–2311.

Zhao, S.Z., *et al.*, 2008. Dynamic multi-swarm particle swarm optimizer with local search for Large Scale Global Optimization. *In*: *2008 IEEE Congress on Evolutionary Computation (CEC 2008).* Hong Kong. 1-6 June. IEEE Press, 3845–3852.

Appendix A. Test problems

The 24 test problems were taken from (Liang *et al.* 2006). Consult the reference for a complete description of them.

Engineering optimization problems:

- **Design of a 10-bar planar truss:** The problem consists of finding the cross-sectional area of each member of the truss, such the weight is minimized subject to both displacement and stress constraints. The weight of the truss is a function $f(x)$ defined in equation (A1). The 10-bar plane truss (Belegundu 1982) adopted is shown in Figure A1.

$$f(x) = \sum_{j=1}^{10} \rho A_j L_j \tag{A1}$$

  where x is the solution proposed, A_j the cross-sectional area, L_j the length of the member j and, ρ the weight density of the material. The following data is assumed:
  - Modulus of elasticity: $E = 1.09 \times 10^4$ ksi.
  - $\rho = 0.10$ lb/in^3.
  - A load of 100 kips in the negative $y-$direction is applied at nodes 2 and 4.
  - Maximum allowable stress of each member: $\sigma_a = \pm 25$ ksi.
  - Maximum allowable displacement of each node (horizontal and vertical): $u_a = 2$ in.
  - Minimum allowable cross-section area for all members: 0.10 in^2.
  - Web thickness and flange thickness are kept fixed at 0.1 in.

  The problem was modeled using 20 design variables, that is, the dimension (height and width) of each element. To evaluate the objective function of this problem, it was necessary to add a module for the analysis of the truss. This module uses the stiffness method (Gere and Weaver 1965) to analyze the structure, and returns the values of the stress and displacement constraints, as well as the total weight of the structure.

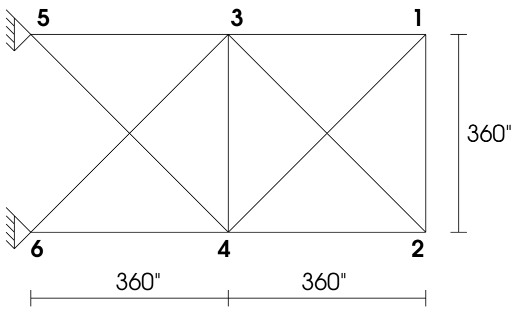

Figure A1. 10-bar plane truss.

Table A1. Node load for 25-bar spatial truss.

Node	F_x (lb)	F_y (lb)	F_z (lb)
1	1,000	-10,000	-10,000
2	0	-10,000	-10,000
3	500	0	0
4	600	0	0

- **Design of a 25-bar spatial truss:** The 25-bar spatial truss adopted is shown in Figure A2. This 3-dimensional problem (Rajeev and Krishnamoorthy 1997) consists of finding the cross-sectional area of each truss member such the weight of the structure is minimized, considering constraints on the maximum allowable displacement of the nodes and on the maximum allowable stress of each bar. The load conditions are shown in Table A1, the coordinates of each node are shown in Table A2 and the groups of elements are shown in Table A3. The following data is assumed:
  - Modulus of elasticity: $E = 1 \times 10^7$ ksi.
  - Material density $\rho = 0.1$ lb/in^3.
  - Maximum allowable stress of each member: $\sigma = \pm 40$ ksi.
  - Maximum allowable displacement: $u = \pm 0.35$ in.

  The problem was modeled using 8 design variables (one for each group), 25 stress constraints and 18 displacement constraints. The weight of the truss is a function $f(x)$ defined in equation (A2).

$$f(x) = \sum_{j=1}^{25} \rho A_j L_j \tag{A2}$$

where x is the solution proposed, A_j the cross-sectional volume, L_j the length of the member j and, ρ the weight density of the material.

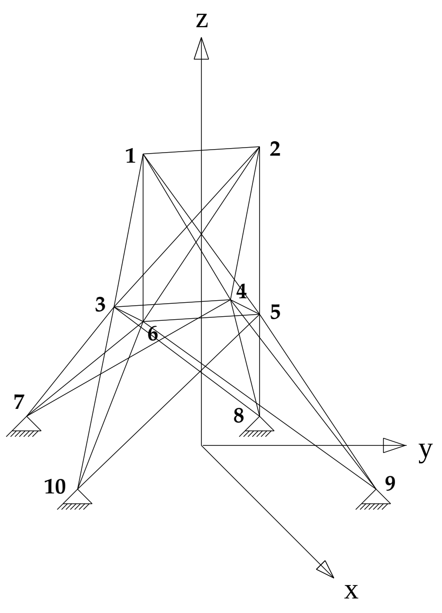

Figure A2. 25-bar spacial truss.

Table A2. Coordinates of nodes for 25-bar spatial truss.

Node	x (cm)	y (cm)	z (cm)
1	-95.25	0	508
2	95.25	0	508
3	-95.25	95.25	254
4	95.25	95.25	254
5	95.25	-95.25	254
6	-95.25	-95.25	254
7	-254	254	0
8	254	254	0
9	254	-254	0
10	-254	-254	0

- **Design of a 200-bar planar truss:** This problem consists of finding the cross-sectional area of each member in the way that its weight is minimized. Displacement and stress constraints have to be considered. The 200-bar plane truss (Belegundu 1982) adopted is shown in Figure A3.

  The following data is assumed:

  - Loading condition 1: 1 kip acting in positive $x-$direction at node points 1, 6, 15, 20, 29, 34, 43, 48, 57, 62 and 71.
  - Loading condition 2: 10 kips acting in negative $y-$direction at node points 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 24, 71, 72, 73, 74 and 75.

Table A3. Groups of members for 25-bar spatial truss.

Group	Member
1	1-2
2	1-4, 2-3, 1-5, 2-6
3	2-5, 2-4, 1-3, 1-6
4	3-6, 4-5
5	3-4, 5-6
6	3-10, 6-7, 4-9, 5-8
7	3-8, 4-7, 6-9, 5-10
8	3-7, 4-8, 5-9, 6-10

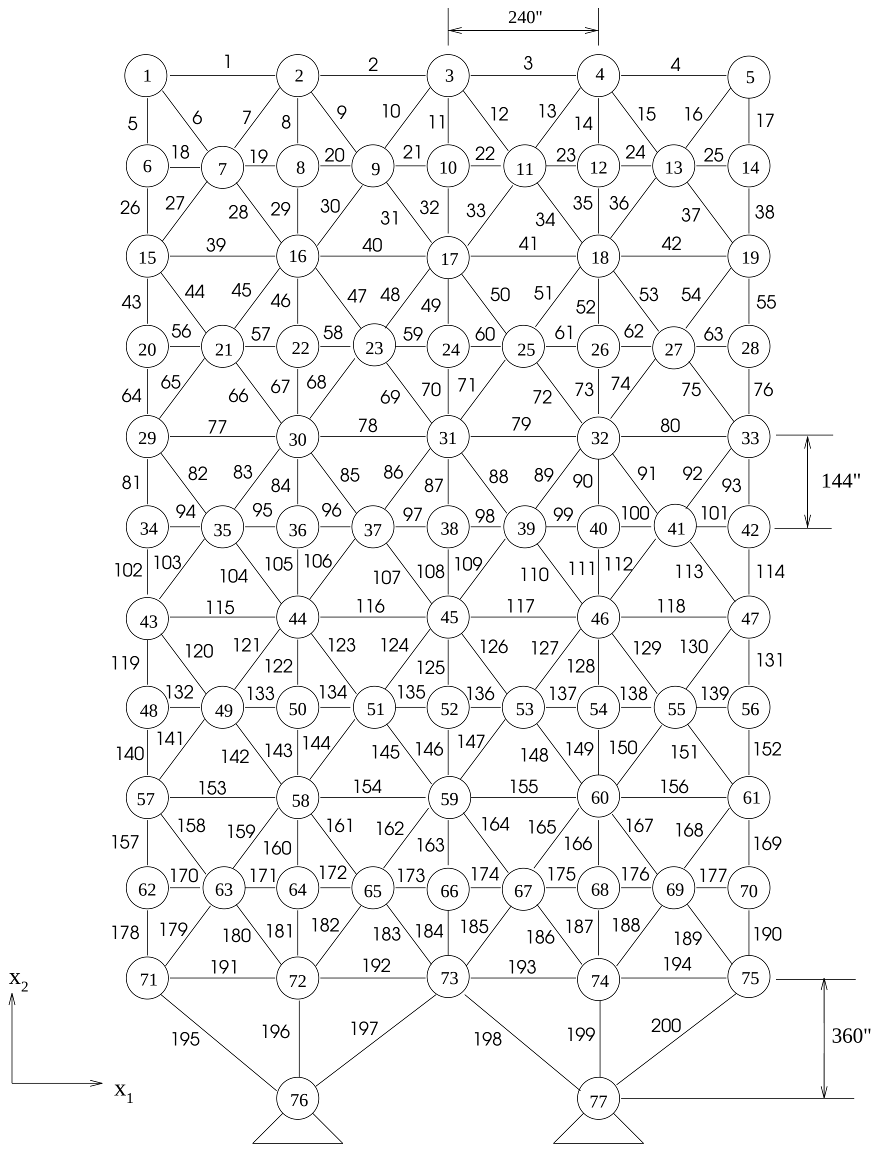

Figure A3. 200-bar planar truss.

Table A4. Groups of members for 200-bar planar truss.

Group	Member
1	1,2,3,4
2	5,8,11,14,17
3	19,20,21,22,23,24
4	18,25,56,63,94,101,132,139,170,177
5	26,29,32,35,38
6	6,7,9,10,12,13,15,16,27,28,30,31,33,34,36,37
7	39,40,41,42
8	43,46,49,52,55
9	57,58,59,60,61,62
10	64,67,70,73,76
11	44,45,47,48,50,51,53,54,65,66,68,69,71,72,74,75
12	77,78,79,80
13	81,84,87,90,93
14	95,96,97,98,99,100
15	102,105,108,111,114
16	82,83,85,86,88,89,91,92,103,104,106,107,109,110,112,113
17	115,116,117,118
18	119,122,125,128,131
19	133,134,135,136,137,138
20	140,143,146,149,152
21	120,121,123,124,126,127,129,130,141,142,144,145,147,148,150,151
22	153,154,155,156
23	157,160,163,166,169
24	171,172,173,174,175,176
25	178,181,184,187,190
26	158,159,161,162,164,165,167,168,179,180,182,183,185,186,188,189
27	191,192,193,194
28	195,197,198,200
29	196,199

- Loading condition 3: loading condition 1 and 2 acting together.
- The 200 elements of this truss are linked to 29 groups (see Table A4).
- Young's modulus of elasticity : 30,000 ksi.
- Weight density: 0.283 $\times 10^3$ kips/in^3.
- Maximum allowable stress of each member (tension and compression): 10 ksi.