A cost-effective algorithm for the solution of engineering problems with particle swarm optimization

Giordano Tomassetti

A cost-effective algorithm for the solution of engineering problems with particle swarm optimization

Journal:	*Engineering Optimization*
Manuscript ID:	GENO-2009-0098.R3
Manuscript Type:	Original Article
Date Submitted by the Author:	24-Aug-2009
Complete List of Authors:	Tomassetti, Giordano; Italian Aerospace Research Centre, Vibro-acoustics and Smart Structures Lab
Keywords:	particle swarm optimization, engineering problems, constrained optimization, nonlinear objective function, nonlinear constraints

A cost-effective algorithm for the solution of engineering problems with particle swarm optimization

Giordano Tomassetti*

Vibro-acoustics and Smart Structures Lab, Italian Aerospace Research Centre, Capua, Italy

A hybrid particle swarm optimization algorithm is proposed for the cost-effective solution of single objective constrained engineering problems. The algorithm implements original strategies aimed to reduce computational effort of optimizations when dealing with real-world problems. Taking inspiration from evolutionary algorithms, a selection mechanism among particles is proposed allowing significant reductions in the solution cost. To enlarge the exploration space, a multi-start approach is frequently adopted, randomly reinitializing the swarm; the injection of optimized particles -obtained in previous runs- into the successive randomly generated starting swarms has been investigated as an expedient to accelerate convergence to the optimal solution. In order to avoid the algorithm to remain trapped into local minima, an innovative scheme has been proposed to update the inertia factor multiplying the previous velocity of the swarm. The proposed algorithm has been validated using standard engineering and purely mathematical problems commonly recognized as valid benchmark functions in specialized literature.

Keywords: particle swarm optimization; constrained optimization; nonlinear objective function; nonlinear constraints.

1. Introduction

In practical engineering optimization problems cost function and constraint evaluations represent one of the most time-consuming aspects of the design. This is because the objective functions or the limitations to the design space are frequently evaluated by evoking some external code (finite elements analyses or other computationally intensive programs). These programs dramatically increase the optimization time inducing the international scientific community to make an effort to develop cost-effective algorithms.

Particle swarm optimization (PSO) is frequently preferred to genetic algorithms (GA) for its intrinsic ability to rapidly drive the design into a minimum. On the other hand, while GA guarantee a deep exploration of the design space, PSO,

* Email: g.tomassetti@cira.it

as originally formulated by Kennedy and Eberhart in 1995 easily falls into local minima. The idea of simulating the graceful choreography of birds in a flock can be mathematically modelled to imitate the psychosocial behaviour of these animals using cooperation to obtain a shared asset. In the original formulation, a swarm of *M* particles –each representing, with its position in the *N*-dimensional hyperspace, one of the possible design parameters- is randomly initialized within the feasible region. For the sake of clarity, throughout the present article the expression ${}^{k}x_i^j$ indicates the i^{th} particle dimension for the j^{th} particle at the k^{th} iteration. At each k^{th} iteration, each j^{th} particle position ${}^{k}\mathbf{X}^j=[{}^{k}x_1^j, {}^{k}x_2^j,\ldots, {}^{k}x_N^j]$ is then updated into ${}^{k+1}\mathbf{X}^j=[{}^{k+1}x_1^j, {}^{k+1}x_2^j,\ldots, {}^{k+1}x_N^j]$ on the basis its own flying experience (the personal best design –*pbest*– the single particle *j* found up to the present k^{th} iteration) and its companions flying experience (the best design –*gbest*– the entire swarm found up to the present k^{th} iteration). The entire swarm **X** made up of volume-less particle is moved according to the following equations:

$${}^{k+1}\mathbf{V} = w\cdot{}^{k}\mathbf{V} + c_1 r_1 \cdot \left(pbest - {}^{k}\mathbf{X}\right) + c_2 r_2 \cdot \left(gbest - {}^{k}\mathbf{X}\right) \quad (1)$$

$${}^{k+1}\mathbf{X} = {}^{k+1}\mathbf{V} + {}^{k+1}\mathbf{X} \quad (2)$$

where *w* is an inertial constant (controls the balance between global and local exploration representing an inertial term to the movement of the individual), c_1 and c_2 are the so-called learning factors representing the weighting of the stochastic acceleration towards the personal best and the global best, respectively. r_1 and r_2 are two random numbers independently generated. In this work, uniformly distributed pseudorandom numbers (obtained using a Mersenne Twister algorithm) are referred to as "random" numbers. The random seed is reinitialized each time a random number is needed.

The most common problem encountered by optimization methods also affects PSO. Despite having demonstrated to be an effective algorithm, with PSO it is quite difficult to identify a global minimum even when the problem strictly requires a global exploration and the detection of a global extreme. Parsopoulos and Vrahatis (2002) proposed a stretching technique performing a two-stage transformation of the cost function to stretch it allowing the optimizer to escape from local minima. On the other hand, to avoid the optimizer to remain trapped into local minima, Battiti *et al.* in 2005 presented the "Affine Shaker algorithm". The term "shaker" refers to the brisk movements of the search trajectories of the local minimizer, while "affine" means the affine transformation executed on the local search region considered for the generation of the successive point for the individual along its trajectory.

In its primordial formulation, PSO did not take into consideration constraints to the design space. However, almost all engineering problems are characterized by a number of constraints often representing the feasibility of the design process. In structural optimization, for example, constraints often express the allowable stresses inside each structural component. Sometimes, when constraint equations stand for the limits for material properties, constraint violation can signify that the optimizer is trying to solve a problem having no physical meaning. When dealing with commercial finite elements codes, this can induce the solver to crash, interrupting the optimization process. Moreover, in complex practical engineering design, optimization is frequently used simply to drive the design into the feasible space and the minimization of the cost function becomes of secondary interest. This usually happens when the designer has already in mind a first-attempt configuration that could be infeasible and needs to re-enter the feasible space. For these reasons, an efficient constraint handling technique is essential for the use of PSO in engineering

optimization. The most common approach in PSO community (as well as for GA, Coello Coello 1999) is to handle constraints using penalty functions. Unfeasible swarms are penalised by adding a term to the fitness. The penalty is proportional to the constraint violation along any dimension of the hyperspace.

The efficiency of any evolutionary algorithm is strongly influenced by its behaviour to terminate a run (Jain *et al.* 2001). In an iterative process, termination criteria influence the effectiveness in identifying a possibly optimal solution to the problem. A large number of criteria have been proposed so far in specialised literature. In practical engineering, a termination criterion should determine the end of a search process as soon as the algorithm is not sufficiently efficient. When the search process degenerates into a random search with no significant improvements, the efficiency of the algorithm is exhausted and further computation only leads to an undesirable solution cost increase (Zielinski and Laur 2007).

Many studies have demonstrated the improvements in performance obtained by hybridizing PSO algorithms implementing some of the principles used for GA. Angeline, in 1998, proposed a tournament selection based on the comparison of each individual's fitness with k other individuals in the swarm. Positions and velocities of the best half of the swarm were used to replace the worst half. In 2008, Grundy and Stacey proposed the implementation of mutation and hill climbing mechanisms in a PSO algorithm obtaining encouraging results in low and moderate dimension problems. On the other hand, van den Bergh and Engelbrecht (2004) demonstrated a significant improvement of the performances by introducing a cooperative behaviour using multiple swarms to optimize different components of the solution vector cooperatively. Tillett *et al.* (2005) investigated whether natural selection can enhance the ability of PSO algorithms to escape from local minima. Richards and Ventura

(2003) studied the effect of swarm size and sociometry on the effectiveness of PSO schemes. Dynamic sociometry proved to be effective in some situations but not all.

2. Dealing with constraints

When dealing with real engineering problems, an efficient approach to constraint handling is crucial to improve the effectiveness of the optimization process. In general terms, any engineering optimization problem can be defined by the following formulation:

$$\min \quad f(\mathbf{X}), where\ \mathbf{X} = \{x_1 \quad x_2 \quad ... \quad x_N\} \in \Re^N \tag{3}$$

$$subject\ to \quad g_h(\mathbf{X}) \le 0, \quad h = 1,2,..., p \tag{4}$$

$$where \quad x_i^L \le x_i \le x_i^U, \quad i = 1,2,..., N \tag{5}$$

Constraints are commonly classified into "geometrical" and "physical" limitations to the design. Geometrical constraints are limitations directly expressed in terms of upper and lower bound of the design variables. Physical constraints are restrictions expressed on physical quantities that need to be numerically evaluated. Real-world limitations are frequently multiple, non-linear and non-trivial constraint functions on the design reducing the feasible space to a small subset of the hyperspace. In these design processes, function and physical constraint evaluations are frequently obtained by evoking one or more external solvers. Constraint violation, both geometrical and physical, may induce the solver to crash or, in some circumstances, to give non-physical results thus leading the optimization process to a numerical solution that has nothing in common with the real problem. For example, this happens when infringing the design space means trying to solve a non-linear structural problem using a linear finite element model.

Consequently, most authors devised a penalty function to estimate the infeasibility level of design candidates. As mentioned before, the basic adopted approach (Coello Coello 1999) is to add a penalty term to the fitness value of infeasible solutions. Though in the author's experience, troubles are frequently encountered in extending this approach to structural optimizations because constraints often require to be strictly honoured. In fact, fitness values in infeasible solutions are sometimes impossible to be evaluated. The most intuitive approach to bypass this obstacle is adding a penalty term $P(^{pres}\mathbf{X})$ for the present candidate $^{pres}\mathbf{X}$ to the last feasible fitness value $f(^{last\ feas}\mathbf{X})$ identified before constraint violation. In mathematical terms:

$$f(^{pres}\mathbf{X}) = \begin{cases} f(^{pres}\mathbf{X}), & if \quad ^{pres}\mathbf{X} \quad feasible \\ f(^{last\ \ feas}\mathbf{X}) + r \cdot P(^{pres}\mathbf{X}), & if \quad ^{pres}\mathbf{X} \quad unfeasible \end{cases} \tag{6}$$

where $^{pres}\mathbf{X}$ is the design vector at the present iteration, $^{last\ feas}\mathbf{X}$ is the last design vector respecting all constraints, r is a multiplying factor set to amplify constraint violation in the penalty evaluation. When constraints are infringed, fitness value at present iteration is calculated summing two terms: fitness value estimated in the last feasible iteration ($f(^{last\ feas}\mathbf{X})$ already known and requiring no further function evaluation) and a penalty $P(^{pres}\mathbf{X})$ expressing the distance from constraints boundaries $[\mathbf{X}^L\ \mathbf{X}^U]$. The penalty function $P(^{pres}\mathbf{X})$ has the following expression:

$$P(^{pres}\mathbf{X}) = \begin{cases} \sum_{i=1}^{N} \left[\max\left(0, x_i^L - {}^{pres}x_i\right)^2 + \min\left(0, x_i^U - {}^{pres}x_i\right)^2 \right], & if \ ^{pres}\mathbf{X} \notin \left[\mathbf{X}^L, \mathbf{X}^U\right] \\ \sum_{j=1}^{p} \min(0, g_j(^{pres}\mathbf{X}))^2, & if \ ^{pres}\mathbf{X} \in \left[\mathbf{X}^L, \mathbf{X}^U\right] \end{cases} \tag{7}$$

The presented approach to constrained optimization has the advantage of avoiding useless -sometimes impossible- cost function evaluation in unfeasible design points

while providing an opportune fictitious value for the fitness to be used by the swarm to move further. In case of constraint infringement, the fitness value is artificially derived to induce the swarm to re-enter the design space as rapidly as possible.

3. Swarm size reduction (SSR)

Taking inspiration from GA, PSO algorithms have frequently been hybridized by manipulating the swarm in order to accelerate the convergence of the individuals to the optimum. The increase in convergence speed is not the only aspect that can be positively affected by swarm modifications. When thinking about complex time-consuming function evaluations, a selection mechanism between particles may be introduced to avoid calculation of fitness in two distinct cases. Figure 1 shows a simplified schematic of a two-dimensional search space with a single constraint (solid line). Dashed curves represent lines along which the objective function is constant. With reference to Figure 1, when an individual of the swarm (bee # 6 in Figure 1) is too far from the presumed global best (the honey jar in Figure 1), probably its elimination from the swarm could be taken into consideration. On the other hand, if two particles are too close to each other (bees # 1 and # 2 in Figure 1), it may be computationally effective to eliminate the worst (bee # 2) of the two allowing only the best one (bee # 1) to proceed towards the global best. Bee # 3 is flying in the infeasible region. So, its fitness is calculated adding a penalty function as explained before. Bees # 4 and # 5 are not close enough to take into consideration the elimination of one of them, despite they have a similar value of the fitness because they lay on the same dashed curve.

On the basis of these practical considerations deriving from common sense, a tournament selection is introduced in the algorithm, allowing the swarm to eliminate

particles thought to give small contribution to the search process. In the commonly used metaphor of bees moving around in a swarm, the selection mechanism simulates the behaviour of insects fighting one another when they get too close to each other looking for the same food. To the author's knowledge of PSO literature, the SSR approach is original while many others have proposed a cooperative approach to the search process (van den Bergh and Engelbrecht 2004). This approach echoes the selection of individuals representing the basis of GA enabling PSO to drastically reduce the computational cost of the algorithm without excessively worsening the quality of the solution.

The consumer theory of microeconomics and the game theory (von Neumann and Morgenstern 1944) can help understanding the inspiration for the SSR methodology. Imagine you are a consumer and you are able to form binding commitments with all the other consumers operating on the same market. In analogy with PSO hypotheses (each particle makes its information available to their neighbours and they are also able to see where their neighbours have had success), it is assumed that communication among consumers is allowed and perfect. This means that any consumer is perfectly informed about any other's choice and he acts, according to a defined strategy, as a consequence of this knowledge. As long as the temporal horizon for purchasing is infinite, the best strategy for each consumer will be to act according to the commitment to maximize the collective benefit. This is analogous to what happens to the particles in the swarm. Particles strictly act according to a social behaviour (a set of rules). Now imagine the temporal horizon of the market is unexpectedly reduced. In the analogy with PSO, this represents the change of the optimization strategy when the number of function evaluations to converge to an improved -even if not optimal- solution has to be reduced. In this case,

consumers will be caught in a bind of having to buy something as soon as possible and they will probably be worried about failing to reach the initially defined communal goal of the swarm. In this case, the most rational choice will probably be to defect from the cooperative strategy preferring an antagonistic behaviour aimed to immediately maximize each own utility. In other terms, the classical cooperative PSO approach is to be preferred when no restrictions are posed on the temporal horizon of the swarm to find the optimum. When the market turns from static (no limits on the number of objective evaluations) into a dynamic one approaching to the closing time (the need for containing the solution cost of the optimization) consumers (i.e. particles) rational choice is to eliminate the less promising candidates giving opportunities only to the most talented ones.

SSR is summarised by the following equation and described in Figure 2:

$$if \begin{cases} \left|x_i^A - x_i^B\right| \le 0.1(x_i^U - x_i^L), i = 1, N \\ \quad and \\ f(\mathbf{X}^A) \le f(\mathbf{X}^B) \quad \Rightarrow del \quad B \\ \quad and \\ k > k_0 \end{cases} \tag{8}$$

$$if \begin{cases} \left|x_i^A - x_i^B\right| \ge 0.4(x_i^U - x_i^L), i = 1, N \\ \quad and \\ f(\mathbf{X}^A) \le f(\mathbf{X}^B) \quad \Rightarrow del \quad B \\ \quad and \\ k > k_0 \end{cases} \tag{9}$$

Equations (8) and (9) simply express the conditions for eliminating one particle (particle *B*, for example). After a predetermined number of iterations k_0, the swarm will probably have already detected a promising movement direction. Preliminary tests showed that $k_0 = 10$ is a reasonable choice. Now, it may happen that two

particles (*A* and *B*) have very similar positions in the search space, as expressed by Equation (8). On the contrary, it may also happen that one (or more) particles are positioned relatively far from the rest of the swarm, as expressed by Equation (9). In both these cases, the SSR allows the swarm to eliminate particles thought to give small contribution to the search process. In fact, when particles *A* and *B* are very close to each other, the elimination of the worst of the two is probably a reasonable choice if the solution cost has severely to be contained. On the opposite, after a number of iterations k_0, it may happen that one particle has got behind the rest of the swarm. In this case, this particle is probably moving in the hyperspace without effectively contributing to the search process. So, a rational option may be to purge it.

In Equations (8) and (9), at each iteration, each particle position is compared to each other in the swarm to identify if a single particle (*B*) is too far from another particle (*A*) or if two particles (*A* and *B*) are too close to one another. Coefficients 0.1 and 0.4 shown in equations (8) and (9) were determined after preliminary tests. When two particles are too far or too close one another, the worst -in terms of fitness- is eliminated from the swarm.

As described in Figure 2, SSR implements a tournament selection based on pairwise comparison between each individual and all the others. The comparisons between couples of particles are performed until exhaustion of individuals. To guarantee a minimal number of individuals in the swarm, SSR applies only as long as the present number of individuals equals at least half of the original population size (initial swarm size). At the end of each iteration, a number of particles is eliminated. So the number of surviving individuals at the end of the optimization can be defined as the final swarm size.

Concerning constraint participation in the selection mechanism, SSR is applied only to individuals respecting both side and physical constraints. This is in order not to interfere with the penalty method which is expressly devoted to guide the search process to reduce constraint violation.

4. Particle injection (PI)

One of the goals of modern heuristic is to combine exploration and exploitation of the search space. The exploration is responsible for the enlargement of the spectra of scanned regions in the hyperspace, while exploitation is devoted to the deep analysis of the most promising subsets of the design space. An ideal optimization algorithm should be able to detect the potentially optimal regions in very few iterations and analyze them thoroughly to efficiently find the global optimum. To increase the algorithm exploration capabilities one can alternatively use more individuals or implement a multi-start process. Both these solutions may compromise the algorithm efficiency causing the solution cost to rise unjustifiably. An interesting strategy could be a multi-start approach treasuring information obtained by previous runs to quickly select the areas to scan more deeply.

When activating the particle injection (PI) subroutine, a particular kind of multi-start approach is implemented (Figure 3). In the ordinary multi-start search, at the beginning of each restarting a new randomly positioned swarm is generated. But previous restarts had already detected a best design candidate. The “injected particle” is the best candidate found during the previous multi-start process, up to the present run. The basic idea of PI technique is to make use of the swarm knowledge of the search-domain. This knowledge is represented by the best design candidate that is “injected” -i.e. directly positioned among the rest of the randomly initialized swarm.

This should guarantee diversity (because of the multi-start approach) and an appropriate use of previous swarm flying experience to speed up the numerical process. As expressed by Equation (10), the best particle is injected in only half of the optimization restarts.

In order to enlarge the number of different scanned subsets of the design space, a common choice is to repeatedly initialize the search process starting from different sets of randomly generated initial particles positions. When activating PI, this is done trying not to disperse the "knowledge" accumulated up to now. The best optimum found among previous numerical search campaigns is considered the best candidate to be a global optimum. Using an ordinary multi-start technique, the flying experience of preceding swarms - that have moved around in the design space in the previous optimization runs - is useless to the present optimization run. On the contrary it could be made available to the present swarm in order to speed up the detection of promising regions without renouncing the advantages given by the multi-start approach in terms of improvements to the exploration capabilities. If a particle, in any preceding optimization runs, finds a promising candidate subset of the hyperspace, the successive swarms will keep the memory of this knowledge and this will hopefully help future swarms to get faster to the optimum.

Obviously, sharing the flying experience among successive generations of swarms could lead to a stagnation of particles in a previously detected promising area thus causing difficulties in exploring new areas. The approach chosen in this study is to use particle injection in only half of the optimization runs in order to avoid the risk of compromising the diversity of swarms, that is, the exploration capabilities of the algorithm. The initialization procedure can be expressed by:

$$\left[{}^{start\ k^{th}\ run}\mathbf{X}\right]=\begin{cases} Rand\left(\left[\mathbf{X}^{j,L},\mathbf{X}^{j,U}\right]\right)\ ,\ with\ \ j=1,..,M,\ \ if\ \ k\ \ odd \\ \begin{cases} X^{1,\ previously\ found\ gbest} \\ Rand\left(\left[\mathbf{X}^{j,L},\mathbf{X}^{j,U}\right]\right)\ ,\ with\ \ j=2,..,M, \end{cases} \quad if\ \ k\ \ even \end{cases} \quad with\ \ \mathbf{X}\in\Re^{N} \qquad (10)$$

where $[{}^{start\ k\ th\ run}\mathbf{X}]$ is the starting swarm made up of M particles for the k^{th} optimization run. If k is an odd number -including the first optimization run when a global best has not been yet detected - the particles positions are randomly generated within the geometrical constraints $[\mathbf{X}^{L}\ \mathbf{X}^{U}]$. If k is an even number -i.e. starting from the second optimization run- one of the starting particles is not a random position vector but it is the "injected" one. This particle will hopefully drive the swarm to rapidly direct to a promising subset of the design space without loosing new regions exploration capabilities because of the randomly initialized rest of the swarm.

5. Inertia weight

An opportune parameters choice is fundamental to increase the overall efficiency of the optimization process. This is particularly true when using PSO based algorithms. In fact, a judicious identification of the parameters can avoid the swarm explosion effect caused by the deleterious effects of randomness as well as the risk of a premature convergence. As previously mentioned, the cognitive and social parameters c_1 and c_2 respectively quantify the memory of previous best position and the neighbours' performance. In this study, these parameters have been considered fixed with the purpose of comparing results obtained here to those of other authors. However, the effect of parameters on the algorithm effectiveness has been investigated and discussed in Section 8.2.

The inertia weight *w* is used to balance global and local search. A large inertia weight facilitates a global search while a small one facilitates a local search. Most authors have then proposed to linearly decrease the inertia weight from a relatively large value to a relative small one (Battiti *et al.* 2005). In this way, the algorithm tends to show a more global exploration capability in the first iteration and a more accurate exploitation aptitude getting closer to the end of the run. Other researchers (Hu *et al.* 2004) have proposed a randomized inertia weight set to *[0.5 + (random/2.0)]*. In the presented algorithm a different approach has been adopted, described by the following:

$$ {}^{k}w = \begin{cases} 0.5 + \dfrac{rand}{2}, & if \quad k = 1 \\ w_1 + (w_2 - w_1) \cdot \dfrac{gbest(k-1)}{gbest(k=1)}, & if \quad k > 1 \end{cases} \tag{11} $$

The inertia weight *w* is firstly initialized according to the randomized approach to *[0.5 + (random/2.0)]*. Starting from the second iteration, ${}^{k}w$ is evaluated according to the ratio between the global best *gbest* at the previous iteration (*k-1*) over the global best *gbest* at the first iteration (*k=1*). w_1 and w_2 are the inertia weight initial and final values. According to Shi and Eberhart (1999), w_1 and w_2 have been respectively fixed to the value 0.9 and 0.4.

There is a reason for preferring this approach to the update process of the inertia weight. In the adopted algorithm, the iteration number is not a correct measure of the number of iterations driving the design to the optimal solution. When one particle infringes the geometrical constraints, its fitness value is fictitiously evaluated as a boundary violation measure. These iterations are accounted for but they do not strictly contribute to the swarm movement towards the optimum because they are only aimed to re-enter the feasible space. Reducing the inertia weight according to these

artificially induced movements did not seem -in the author's judgement- a concrete step further in the search process. A valid alternative to quantify the optimum approaching was identified in measuring how different from the first iteration global best is from the previous iteration global best. In other terms, instead of reducing w according to the increase of iteration numbers, w is reduced on the basis of the decrease of the most recent global best with respect to the first iteration one. Of course, the described approach is appropriate for minimization problems but the reader can readapt it to maximizations by simply inverting the ratio between the global bests in Equation (11).

6. Numerical experimental setting

In order to evaluate pros and cons of the presented algorithm, four engineering design problems have been chosen and two types of tests have been performed. The four engineering benchmark problems have been solved trying to reproduce the settings described by other researchers in previously published experiments. All selected benchmark problems are characterised by nonlinear objective functions and/or nonlinear constraints.

The two sets of tests were aimed to evaluate the solution quality and the convergence speed of the proposed algorithm. Solution quality is measured in terms of difference between the obtained results and the best known optimal values of the objective functions for the four benchmark problems. Convergence speed is measured comparing the number of objective function evaluations needed to reach a certain threshold of fitness with values available in literature.

A brief description of the four benchmark engineering problems follows.

6.1 Design of a pressure vessel

This benchmark problem takes into consideration the design of a compressed air storage tank (Figure 4) working at a pressure of 3,000 *psi* (2.07×10^7 *Pa*) with a minimum volume of 750 ft^3 (21.2 m^3). The tank is capped at both ends with two hemispherical heads. The objective is to minimize the total cost, including forming the welding. Design variables are: x_1 thickness of the shell, x_2 thickness of the head, x_3 the inner radius and x_4 the length of the cylindrical portion of the vessel. x_1 and x_2 are integer multiples of 0.0625 *in* (0.15875 *cm*, the available thickness of rolled steel plates) while x_3 and x_4 are continuous variables.

The problem is stated as:

Minimize:

$$f(X) = 0.6224 \cdot x_1 x_3 x_4 + 1.7781 \cdot x_2 x_3^{\,2} + 3.1661 \cdot x_1^{\,2} x_4 + 19.84 \cdot x_1^{\,2} x_3 \tag{12}$$

Subject to the following physical constraints:

$$\begin{cases} g_1(X) = -x_1 + 0.0193 \cdot x_3 \le 0 \\ g_2(X) = -x_2 + 0.00954 \cdot x_3 \le 0 \\ g_3(X) = -\pi x_3^{\,2} x_4 - 4/3 \cdot \pi x_3 + 1{,}296{,}000 \le 0 \\ g_4(X) = x_4 - 240 \le 0 \end{cases} \tag{13}$$

The geometrical constraints are as follows: $1 \times 0.0625 \le x_1$ and $x_2 \le 99 \times 0.0625$, $10.0 \le x_3$ and $x_4 \le 200.0$. When dealing with integers, x_1 and x_2 are truncated to integers, as done by Hu *et al.* (2003). Although discrete problems have recently been addressing interests in the scientific community, a rigorous implementation of integer variables in PSO algorithms still represents a challenge by itself (Garcia and Perez 2008) and it is not among the aims of this article. Furthermore, even if this is not to be considered the

best possible approach, truncation to the nearest integer value has been chosen by all the authors this article compares its results to (Cagnina 2008, Hu 2003, Coello 1999, Akhtar 2002) in Section 8.

6.2 Welded beam design

The second benchmark example deals with a typical engineering design problem originally described by Deb in 1991 as benchmark test for optimization algorithms. Minimize the fabrication cost of a welded beam (Figure 5) subject to constraints on shear stress τ, bending stress σ, buckling load P_c and end deflection δ. The four design variables x_1, x_2, x_3 and x_4 are all continuous and represent the geometrical parameters of the bar (respectively h, l, t and b).

The optimization problem is summarised as:

Minimize:

$$f(X) = 1.10471 \cdot x_1^2 x_2 + 0.04811 \cdot x_3 x_4 (14.0 + x_2) \tag{14}$$

Subject to the following physical constraints:

$$\begin{cases} g_1(X) = \tau(X) - \tau_{MAX} \le 0 \\ g_2(X) = \sigma(X) - \sigma_{MAX} \le 0 \\ g_3(X) = x_1 - x_4 \le 0 \\ g_4(X) = 0.10471 x_1^2 + 0.04811 x_3 x_4 (14.0 + x_2) - 5.0 \le 0 \\ g_5(X) = \delta(X) - \delta_{MAX} \le 0 \\ g_6(X) = P - P_C(X) \le 0 \end{cases} \tag{15}$$

where:

$$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''(x_2/2R) + (\tau'')^2}$$
$$\tau' = P/(\sqrt{2}x_1x_2)$$
$$\tau'' = MR/J$$
$$M = P(L + x_2/2)$$
$$R = \sqrt{x_2^2/4 + (x_1 + x_3)^2/4}$$
$$J = 2\{\sqrt{2}x_1x_2[x_2^2/12 + (x_1 + x_3)^2/4]\}$$
$$\sigma(X) = 6PL/x_4x_3^2$$
$$\delta(X) = 4PL^3/Ex_4x_3^3$$
$$P_c(X) = (4.013E/L^2)\cdot\sqrt{x_3^2x_4^6/36}\cdot(1 - (x_3/2L)\sqrt{E/4G}) \quad (16)$$

and the following parameters are used: P = 6000 *lb* (26,689 *N*), L = 14 *in* (356 *mm*), $E = 30 \times 10^6$ *psi* (2.068×10^{11} *Pa*), $G = 12 \times 10^6$ *psi* (8.27×10^{10} *Pa*), τ_{MAX} = 13,600 *psi* (9.38×10^7 *Pa*), σ_{MAX} = 30,000 *psi* (2.07×10^8 *Pa*), δ_{MAX} = 0.25 *in* (0.635 *cm*).

The side constraints for the design variables are expressed by: $0.125 \le x_1 \le 2.0$, $0.1 \le x_2 \le 10.0$, $0.1 \le x_3 \le 10.0$, $0.1 \le x_4 \le 2.0$.

6.3 Weight of a tension/compression spring

The minimization of the weight of a tension/compression spring (Figure 6) subjected to constraints on minimum deflection, shear stress, surge frequency is mathematically described as follows.

Minimize:

$$f(X) = (x_3 + 2)x_2x_1^2 \quad (17)$$

subject to:

$$\begin{cases} g_1(X) = 1 - x_2^{\,3} x_3 / 71{,}785 x_1^{\,4} \le 0 \\ g_2(X) = \left(4x_2^{\,2} - x_1 x_2\right) / 12{,}566\left(x_2 x_1^{\,3} - x_1^{\,4}\right) + 1/5{,}108 x_1^{\,2} - 1 \le 0 \\ g_3(X) = 1 - 140.45 x_1 / x_2^{\,2} x_3 \le 0 \\ g_4(X) = \left(x_2 + x_1\right)/1.5 - 1 \le 0 \end{cases} \tag{18}$$

in the optimization process $x_1 = d$ (the wire diameter), $x_2 = D$ (the mean coil diameter), $x_3 = N$ (the number of active coils). Side constraints are: $0.05 \le x_1 \le 2.0$, $0.25 \le x_2 \le 1.3$, $2.0 \le x_3 \le 15.0$.

6.4 Speed reducer

The weight of a speed reducer (Figure 7) is minimized with constraints on bending stress of gear teeth, surface stress, transverse deflections of the shafts and stresses in the shafts (Golinski 1973). This example is reported by other researchers to challenge various optimization algorithms. Design variables are: face width (x_1), module of teeth (x_2), number of teeth in the pinion (x_3), length of the first shaft between bearings (x_4), length of the second shaft between bearings (x_5), diameter of the first (x_6) and second shaft (x_7). All design variables are continuous with the exception of x_3 which is truncated to nearest integer value. The optimization problem is stated as follows:

Minimize:

$$f(X) = 0.7854 x_1 x_2^{\,2}\left(3.3333 x_3^{\,2} + 14.9334 x_3 - 43.0934\right) - 1.508 x_1\left(x_6^{\,2} + x_7^{\,2}\right) + \\ + 7.4777\left(x_6^{\,3} + x_7^{\,3}\right) + 0.7854\left(x_4 x_6^{\,2} + x_5 x_7^{\,2}\right) \tag{19}$$

subject to:

$$
\begin{cases}
g_1(X) = 27/x_1 x_2^{\,2} x_3 - 1 \le 0 \\
g_2(X) = 397.5/x_1 x_2^{\,2} x_3^{\,2} - 1 \le 0 \\
g_3(X) = 1.93 x_4^{\,3}/x_2 x_3 x_6^{\,4} - 1 \le 0 \\
g_4(X) = 1.93 x_5^{\,3}/x_2 x_3 x_7^{\,4} - 1 \le 0 \\
g_5(X) = -1 + \sqrt{\left(745 x_4/(x_2 x_3)\right)^2 + 16.9\times 10^6}/110.0 x_6^{\,3} \le 0 \\
g_6(X) = -1 + \sqrt{\left(745 x_5/(x_2 x_3)\right)^2 + 157.5\times 10^6}/85.0 x_7^{\,3} \le 0 \\
g_7(X) = x_2 x_3/40 - 1 \le 0 \\
g_8(X) = 5 x_2/x_1 - 1 \le 0 \\
g_9(X) = x_1/12 x_2 - 1 \le 0 \\
g_{10}(X) = \left(1.5 x_6 + 1.9\right)/x_4 - 1 \le 0 \\
g_{11}(X) = \left(1.1 x_7 + 1.9\right)/x_5 - 1 \le 0
\end{cases}
\tag{20}
$$

and with the following side constraints: $2.6 \le x_1 \le 3.6$, $0.7 \le x_2 \le 0.8$, $17 \le x_3 \le 28$, $7.3 \le x_4 \le 8.3$, $7.8 \le x_5 \le 8.3$, $2.9 \le x_6 \le 3.9$, $5.0 \le x_7 \le 5.5$.

7. Solution quality: numerical results and discussion

Solution qualities of the optimizations performed with the proposed algorithm have been quantified by comparing the obtained solutions with the best optima found by different authors. Conditions described by Hu *et al.* (2003) -briefly referred to as Hu in the following- have been considered as the experimental set up for solution quality tests. According to this article, the swarm size is 20, maximum generation is 10,000, in all the experiments. Differently from Hu, the initial population is not repeatedly initialized until all randomly positioned particles meet all the constraints. In fact, this approach only caused a numerical explosion of the solution cost without giving significant improvement to the solution quality. However, not having initialized the swarm to the feasible space should represent a pejorative condition for the presented algorithm. Nevertheless, this aspect has proved not to be a real problem to the solution

quality nor to the convergence speed. According to Hu, the learning rates c_1 and c_2 were set to 1.49445 while the maximum speed $\mathbf{V}_{max}$ was set to the dynamic range of the particle along each dimension. As described by Hu, eleven runs were executed for each of the four problems and the best solution for each is reported in the following tables (Table 1 to Table 4). Optimal design variables, constraints in the optimum and the optimal objective are indicated and compared to the best solutions found in literature. To the end of faithfully reproducing Hu's experimental conditions, the particle swarm size reduction (SSR) and particle injection (PI) have been deactivated for these runs. Nevertheless, very similar results were obtained by activating the mentioned SSR and PI subroutines of the algorithm but they are omitted for the sake of brevity. Results obtained with the proposed algorithm were also compared with a number of mathematical test functions -namely Ackley's function, Grienwagk's function, De Jong's sphere function and the Alpine function (Molga and Smutnicki 2005)- giving acceptable results. These results are omitted since the emphasis of this work is focused on the reduction of engineering problems solution cost. However, a statistical analysis of the convergence speed for the mathematical test functions is reported in Section 8.1.

The comparison of the objective functions in the optima (Table 1 to Table 4) proves the presented algorithm's capabilities to closely approach the best known optima available in literature. In all the four benchmark problems, the differences between the optima found in this work and in literature are negligible.

Some statistics about the objective function is reported in Table 5. For each row -i.e. each benchmark problem- Table 5 shows the mean, the worst and the standard deviation for the optimal values found over eleven runs. As mentioned before, best objective functions over eleven runs are reported in Tables 1 to 4 with

details about design variables and constraints in the optima. Table 5 also shows statistics reported by Coello Coello (1999) with respect to the first three reported engineering problems.

Table 5 shows an improvement in the mean optimal objective values with respect to Coello Coello's (1999). Nevertheless, the higher standard deviation values seem to indicate that the implemented algorithm is less robust than Coello Coello's.

The results (from Table 1 to Table 4) coincide with the best ones available in literature allowing to consider the proposed algorithm (integrated or not with SSR and PI subroutines) as a validated PSO optimizer to face the challenging aspect of increasing the convergence speed when dealing with real-world optimization problems.

8. Convergence speed: numerical results and discussion

In the frame of this study, the convergence speed of the algorithm can be simply defined as the rapidity for the objective function to slope down a specified threshold. The rapidity is expressed in terms of number of objective functions evaluations and compared to data available in specialised literature.

The importance of convergence speed in solving real engineering problems -as mentioned in the preceding sections- is strictly connected with the severe restrictions posed by a large number of problems requiring long computational time for each objective calculation. An almost immediate improved solution to a real problem -even if it's not the best possible- is often preferred to a more accurate, possibly optimal solution available after an unsustainable CPU time.

Literature review revealed a scarcity of data about convergence speed -as it has just been defined- for engineering problems. Most articles are centred on the improvement of the solution quality.

Cagnina *et al.* (2008) and others referred to Akhtar *et al.* (2002) -briefly referred to as Akhtar in the following- and compared the four presented benchmark problems solution quality. Cagnina definitely improved the solutions for these problems and obtained statistics about them by 30 independent runs per problem with 24,000 function evaluations per run.

Since the mentioned Akhtar's article is one of the few presenting data about the number of cost function evaluations for the four selected problems, the basic approach to estimate the convergence speed was to iterate the optimization process until Akhtar's optimal objective function values are obtained. The solution cost is then compared with the values reported by the same author (Table 6). In other terms, the convergence criterion is substituted by a termination condition based on the achievement of the optimal solutions reported by Akhtar.

Since the comparison for the convergence speed is based on Akhtar's data, the experimental conditions described by this author were faithfully reproduced. First of all, only three out of the four benchmark problems have been compared because Akhtar did not analyse the tension/compression spring problem. Of course, it was impossible to exactly reproduce the specifications described by Akhtar because he proposed a social interaction model among society leaders that is quite different from the scheme adopted in the present article. However, as other authors did, Akhtar's solution cost can be used as a valid reference. To imitate his numerical specifications, 10 multi-start tests were performed for each benchmark problem and the execution was stopped as soon as the fitness reached 2.4426 for the welded beam design,

3,008.08 for the speed reducer and 6,171.00 for the pressure vessel. Each single run of the 10 multi-start tests was stopped as the fitness fell below the respective threshold or the number of iterations exceeded 100 (Figure 8). In case the single run was terminated because of the achievement of the desired fitness threshold, even the respective multi-start test was terminated and the solution cost was accounted for. Otherwise, if the single run terminated for iteration overflow, a restart of it was evoked. In this last case, the continued search process could take advantage or not of the best optimum found up to that moment, depending whether the PI subroutine was activated or not. Obviously, in case of restarting, all the preceding runs solution costs were summed to the last one to determine a total solution cost. At the end of the 10 multi-start tests, the average cost function evaluations number was calculated and compared with Akhtar's values. For the sake of precision, the obtained solution costs were also compared with those found calculating the average over 100 runs. A convenience in implementing the SSR was recorded as well. Solution costs averaged over 100 runs have been discussed in Section 8.1 and 8.2 where PSO parameters and swarm size effects have been investigated (Table 8 and 9).

The comparison was made by alternatively activating and deactivating the SSR and the PI subroutines (see the different columns of Table 6) in order to isolate the two proposed techniques effects on the algorithm convergence speed.

Table 6 shows a significant reduction of the average number of objective functions evaluations with reference to Akhtar's. For each of the three benchmark problems analysed by the mentioned author, the presented algorithm proved to be able to increase the convergence speed. Even if it was not considered in Akhtar's article, the fourth problem –the tension/compression spring problem- gave analogous results. Of course different tendencies are highlighted by each benchmark problem. To call

attention to the lower solution costs in Table 6 for each row, these values have been bolded. Giving a short look at Table 6, the reader may notice a concentration of bold numbers -*i.e.* lower solution costs- in the first two columns (SSR+PI activated and SSR-only activated).

For the welded beam design problem, the proposed algorithm yields the best results since the threshold of 2.4426 is reached in about 8% (at the most) of the iterations necessary to Akhtar. For this benchmark problem, the best technique is to activate only the SSR subroutine. This gives an average solution cost of 881.3 vs. 19,259 experienced by Akhtar.

The design of a pressure vessel problem gives an average solution cost (11,721) comparable to the reference value by Akhtar (12,630) when both the SSR and the PI subroutines are deactivated. A certain improvement was obtained by activating only the SSR (10,527) but the best average value is achieved when using PI only (7,802) or both SSR and PI (8,445). On the contrary to what was expected, the contemporary use of both SSR and PI causes a slight worsening of the results.

Although Akhtar did not consider the tension/compression spring problem, it is interesting to notice that it gave an average solution cost of 8,139 with SSR and PI both deactivated. The termination criterion to stop the runs and count the number of objective functions evaluations was based on the optimal value of 0.0127 (i.e. about 0.27% more than the best minima found in literature, see Table 3). For this benchmark problem the best average performance is obtained by using PI only (5,653). However, this value is not so far from the one achieved by activating both SSR and PI (6,094).

For the speed reducer design problem, the algorithm achieves the best average attainment when both SSR and PI are active (5,420 vs. 19,154 by Akhtar). Similar

results were obtained when only the SSR is active (5,458). A relative deterioration was recorded when activating PI only or deactivating both SSR and PI.

In order to give statistical significance to the comparison between solution costs reported in Table 6, standard deviations, minima and maxima are displayed for each benchmark problem. Despite the scarcity of rigours statistical analyses about engineering problems solution costs in technical literature, some considerations about data reported in Table 6 can be done. From the analysis of Table 6, being standard deviation commonly used to measure confidence in statistical means, one may conclude that the dispersion of the statistical population confirms an “acceptable” tendency of data points to be close to the mean value. It has to be underlined an intrinsic difficulty in defining what is “acceptable” without any possible comparison to analogous works in specialised literature.

Also from the analysis of minima and maxima in Table 6, the reader may easily notice a similar trend to the one described for the mean values. In general terms, a convenience in the contemporaneous use of SSR and PI or in the use of SSR only can be clearly identified.

With the unique exception of the welded beam design benchmark problem, the lowest standard deviation is always detected in correspondence to the SSR+PI or the only-SSR activated columns. But it has to be noticed how close the standard deviation for the last column of the welded beam is close to the one in the first column. It may be concluded that both SSR+PI and SSR-only are better clustered closely around the mean. This may be interpreted as a proof of robustness of the proposed SSR technique.

8.1 Convergence speed tests for mathematical benchmark functions

Most of the times, optimization engineers are required not to determine their real-world problem global optimum taking days of calculation but they are asked to find an improved configuration in a reasonable CPU-time. From an engineering point of view, a non-relaxable constraint is always represented by time spent for computations. Most precisely, this limitation should be expressed in terms of a desirable restriction to the number of cost function evaluations since this is synonymous with the optimization economic cost. In this frame, the present article proposes the SSR and PI techniques expressly devoted to the *cost-effective* solution of engineering optimization problems. It is a matter of fact that a relatively small improvement in the solution after hundreds of iterations, may have no importance at all, from an industrial point of view. On the contrary, when one switches his/her point of view to evolutionary computing, algorithmic performances are obviously analysed overall, sometimes even after hundreds (or thousands) of iterations.

Nevertheless, a statistical analysis of results obtained for mathematical test functions may be interesting to evaluate the effectiveness and efficiency of the proposed algorithm. To do this, the numerical approach proposed in many works to compare plots representing the cost function vs. the iteration number and consequently discuss the results may be adopted. But this may be not coherent with the aims of this article because no direct stress is given on solution costs.

Four mathematical test functions (Molga and Smutnicki 2005) have been selected (Ackley's, Griewangk's, Sphere and Alpine functions) and an objective function threshold has been fixed as termination criterion.

Ackley's function is a multimodal test function having the following expression:

$$f(x) = -a \cdot \exp(-b \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n} \cos(cx_i)) + a + \exp(1) \quad (21)$$

where $a = 20$, $b = 0.2$, $c = 2\pi$. Search area is restricted to the subset $-32 \leq x_i \leq 32$. It has a global minimum $f(x) = 0$ for $x_i = 0$, $i = 1,\ldots,n$.

Griewangk's function has the following definition:

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1 \quad (22)$$

Search domain is usually $-500 \leq x_i \leq 500$ and this function has a global minimum $f(x) = 0$ for $x_i = 0$, $i = 1,\ldots,n$.

The sphere function is defined as follows:

$$f(x) = \sum_{i=1}^{n} x_i^2 \quad (23)$$

Search area is generally restricted to hypercube $-5.12 \leq x_i \leq 5.12$ and a global minimum $f(x) = 0$ is located in $x_i = 0$, $i = 1,\ldots,n$.

The Alpine function is defined as:

$$f(x) = \sum_{i=1}^{n} \left| x_i \cdot \sin(x_i) + 0.1 \cdot x_i \right| \quad (24)$$

Search space is commonly identified by the hypercube $-10 \leq x_i \leq 10$ and a global minimum $f(x) = 0$ is located in $x_i = 0$, $i = 1,\ldots,n$.

Results are shown in Table 7 in terms of statistics for the solution cost (necessary number of cost function evaluations to reach this threshold). In a similar way to that pursued for the engineering test problems (Figure 8), 100 multi-start runs

were performed for each considered benchmark function and the execution was stopped as the fitness reached the values reported in the second column of Table 7. When not terminated for cost function threshold reached, each single run -of the 100 multi-start tests- was alternatively interrupted in case of iteration overflow. This happened as the number of iterations exceeded 500 (Figure 8). In Table 7, lower solution costs for each row have been bolded.

The reader may notice termination criteria displayed in Table 7 are not excessively severe. The reason for this choice is that the attention here is mainly focused on *fitness decreasing rate* with respect to the solution cost. Even a local minimum detection may be acceptable as long as the calculation is "fast" enough. Optimization engineers may always reserve the opportunity to repeat the calculations -using a multi-start approach, for example- to enlarge the explored areas if a larger amount of resources is made available for the optimization phase.

From the analysis of Table 7, the reader may easily notice the effectiveness of the SSR technique on the alleviation of the computational effort of PSO. In particular, the use of SSR only, seems to give more efficiency to the search process avoiding unnecessary objective function evaluations. Lower standard deviation values are concentrated on the SSR column to indicate the robustness of this technique implemented within the PSO algorithm.

Table 7 also shows a deterioration in the solution costs when using PI only. This may be caused by a stagnation of the search process in previously determined local optima because of the injected particle or by a non-optimal choice of the max iteration number to restart the single numerical test (Figure 8). This problem becomes evident when dealing with mathematical test functions with a large number of local minima. At the moment, this aspect still needs further investigations.

8.2 PSO parameters and swarm size effect on algorithmic performances

Learning factors c_1 and c_2 strongly influence PSO algorithmic behaviour since they weight the stochastic movements towards the personal best and the global best, respectively. Many different settings have been proposed recently (Cui *et al.* 2008) considering variations in the coefficients to speed-up the search process. Nevertheless, in the great majority of works about PSO, social and cognitive learning factors are chosen to be fixed and have the same value.

Table 8 illustrates the effect of c_1 and c_2 variations on the solution costs for the four engineering benchmark problems. Adopting the same numerical procedure represented in Figure 8, averaged solution cost and standard deviation for $c_1 = c_2$ varying from 1.00 to 2.00 are indicated in Table 8. In Table 8, mean values and standard deviations are calculated over a population of 100 multi-start tests. To give statistical significance to the comparison, values referred to $c_1 = c_2 = 1.49445$ (already displayed in Table 6 averaging over ten multi-start tests to reproduce Akhtar's numerical specifications) are recalculated over 100 multi-start tests. Even in Table 8, lower solution costs for each row have been bolded.

From Table 8, the reader may easily observe a deterioration of the algorithm effectiveness when PSO parameters are switched to $c_1 = c_2 = 1.00$. In general terms, when social and cognitive factors are reduced to 1.00, an increase in the computational effort is registered for all the benchmark test problems.

Parameters increase to $c_1 = c_2 = 2.00$ caused a positive effect on the welded beam design and the speed reducer problems. In these cases, a decrease of the number of calculations may be perceived. However, this effect is opposed when one focuses his/her attention on the other two benchmark problems.

Once more, it has to be stressed how the lack of statistical analyses about engineering problems solution costs in specialised literature makes difficult to judge about the standard deviation. However, comparing the ratio between the standard deviations and the averaged solution costs for the different cases, c_1 and c_2 variations seems to have no significant impact on algorithm robustness.

Richards and Ventura (2003) studied the effect of swarm size on PSO effectiveness for a number of mathematical test functions concluding that larger swarms tend to be more effective on functions having more numerous local minima. In their study, the population size ranged from 5 to 60 individuals and performance were measured in terms of solution quality.

In Table 9, solution costs have been compared for 100 multi-start tests with 20, 50 and 100 individuals respectively to analyse the effect of swarm size on algorithm effectiveness. In this numerical experiment, PSO parameters have been fixed to values $c_1 = c_2 = 1.49445$. Values reported for a swarm size of 20 particles are the same displayed in Table 8 (for the case $c_1 = c_2 = 1.49445$) and are here displayed again to facilitate the comparison with the other cases. Numerical test procedure is explained in Figure 8 and it is the same already used for the other solution cost experiments. To give more emphasis on lower solution costs for each row reported in Table 9, these values have been bolded.

From the examination of Table 9, a clear trend in increasing the solution costs while increasing the swarm size is noticeable for both the design of a pressure vessel and the tension/compression spring problems. On the contrary, the welded beam design shows an opposite trend: solution costs decrease as the swarm becomes larger. It is not easy to judge about the effect of swarm size on the speed reducer problem since it seems to have no significant effect when SSR subroutine is active. On the

opposite, Table 9 shows a decrease in solution cost as the swarm size increases for the mentioned benchmark problem in case the SSR is not in use.

Similar tendency effects are registered in evaluating the influence of the number of particles in the swarm on the standard deviations.

9. Conclusions

In this article original methodologies have been proposed for the aim of reducing the computational effort of real-world nonlinear engineering optimization problems. The results for all the considered benchmark problems demonstrated the effectiveness of the proposed techniques to significantly reduce the number of function evaluations to approach the minima.

By itself, the simple concept of using a fictitious objective -given by the last feasible calculated objective plus a penalty function- avoids an unnecessary solution cost increase. Besides, this approach also avoids the analysis of possibly meaningless physical problems preventing computation when the design is out of geometrical boundaries.

Exploration capabilities of PSO algorithms are frequently improved by repeating the optimization process more than once starting from randomly generated particles positions (multi-start approach). To the aim of reducing the solution cost, this aspect suggested to inject one particle in the best position found up to the present run in half of the successive optimization runs. The PI methodology represents an acceptable compromise between the need for new areas exploration and the desirable solution cost containment when dealing with real-world engineering problems. Further investigation is needed to implement PI for mathematical test functions characterized by a large number of local minima.

Taking inspiration from GA, a selection mechanism among particles (SSR) was introduced. This technique is inspired by the competition among individuals searching for a common food source. As explained by Economics, when resources are limited –a restricted number of optimal solutions- and the market moves into the closing phase -the necessity of containing the solution cost- consumers (*i.e.* the particles) generally abandon any cooperation strategy (that is flying in swarm) fighting each other to maximize their own satisfaction.

References

Angeline, P.J., 1998, Using selection to improve particle swarm optimization, *In: Evolutionary Computation Proceedings,* 4-9 May 1998, Anchorage, AK, USA.

Akhtar, S., Taj, K., Ray, T., 2002, A socio-behavioural simulation model for engineering design optimization, *Engineering Optimization*, 34 (4), 341 - 354.

Battiti, R., Brunato, M., Pasupuleti, S., 2005, *Do not be afraid of local minima: affine shaker and particle swarm* [online], Università degli Studi di Trento. Available from: http://dit.unitn.it/~brunato/pubblicazioni/DIT-05-049.pdf [Accessed 23 April 2009].

Cagnina, L., Esquivel, S., Coello Coello, C., 2008, Solving engineering optimization problems with the simple constrained particle swarm optimizer, *Informatica,* 32 (3), 319-326.

Coello Coello, C., 2000, Use of a self-adaptive penalty approach for engineering optimization problems, *Computers in Industry*, 41 (2), 113 - 127.

Cui, Z., Zeng, J., Yin, Y., 2008, An improved PSO with time-varying accelerator coefficients. *In: Eigth International Conference on Intelligent Systems Design and Applications*, November 26-28, 2008, Kachsiung, Taiwan.

Deb, K., 1991, Optimal design of a welded beam via genetic algorithms, *AIAA Journal*, 29 (11) 2013-2015.

Garcia, F., Perez, J., 2008, *Jumping Frogs Optimization: a new swarm method for discrete optimization* [online]. Available from: http://webpages.ull.es/users/estinv/Investigacion/pdfs_dt/DT_DEIOC_3_2008.pdf [Accessed 22 June 2009].

Golinski, J., 1973, An adaptive optimization system applied to machine synthesis, *Mechanism and Machine Synthesis*, 8, 419-436.

Grundy, I.H., Stacey, A., 2008, Particle swarm optimization with combined mutation and hill climbing, *Complexity International*, 12.

Hu, X., Eberhart, R. C., Shi, Y., 2003, Engineering optimization with particle swarm. *In: Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, April 24-26, 2003, Indianapolis, Indiana, USA.

Hu, X., Shi, Y. Eberhart, R.C., 2004. Recent advances in Particle swarm, *In: Proceedings of the IEEE Congress on Evolutionary Computation*, 19-23 June 2004, 0-7803-8515-2/04/$20.00©2004 IEEE.

Jain, B. J., Pohlheim, H. and Wegener, J., 2001., On Termination Criteria of Evolutionary Algorithms. *In*: *in Spector, L. (ed.): GECCO'2001 - Proceedings of the Genetic and Evolutionary Computation Conference*, San Francisco, CA: Morgan Kaufmann, 768.

Molga, M., Smutnicki, C., 2005, *Test functions for optimization needs* [online]. Available from: http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf [Accessed 23 April 2009].

Parsopoulos, K.E., Vrahatis, M.N., 2002, Recent approaches to global optimization problems through particle swarm optimization, *Natural Computing,* 1, 235-306.

Richards, M., Ventura, D., 2003, Dynamic Sociometry in Particle Swarm Optimization, *In: International Conference on Computational Intelligence and Natural Computing,* September 26-30, 2003*,* Cary, North Carolina USA.

Shi, Y., Eberhart, R.C., 1999, Empirical study of particle swarm optimization, *In: Proceedings of the 1999 Congress on Evolutionary Computation*, 0-7803-5536-9/99/$10.00©1999 IEEE.

Tillett, J., Rao, T.M., Sahin, F., Rao, R., 2005, Darwinian Particle Swarm Optimization, *In*: *Proceedings of the 2nd Indian International Conference on Artificial Intelligence*, December 20-22, 2005 Pune, India.

van den Bergh, F.; Engelbrecht, A.P., 2004, A Cooperative approach to particle swarm optimization, *Evolutionary Computation, IEEE Transactions on,* 8 (3), 225 - 239.

von Neumann, J., Morgenstern, O., 1944, *Theory of Games and Economic Behavior*, Princeton University Press.

Yin, P., Yu, S., Wang, P., Wang, Y., 2006, A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems, *Computer Standards & Interfaces*, 28 (4), 441-450.

Zielinski, K., Laur, R., 2007, Stopping criteria for a constrained single-objective particle swarm optimization algorithm. *Informatica,* 31 (1), 51-59.

Figure captions

Figure 1. Schematic of the selection mechanism among particles (SSR).

Figure 2. Schematic diagram of the SSR technique.

Figure 3. Schematic diagram of the PI technique.

Figure 4 The pressure vessel design problem.

Figure 5. The welded beam design problem.

Figure 6. The tension/compression spring design problem.

Figure 7. The speed reducer design problem.

Figure 8. Schematic diagram of the numerical procedure for solution cost tests.

Table 1. Comparison of the solution quality for the welded beam design problem.

	This paper	Cagnina (2008)	Hu (2003)	Coello (1999)	Deb (1991)
x_1 (h)	0.205729	0.205729	0.20573	0.2088	0.2489
x_2 (l)	3.470489	3.470488	3.47049	3.4205	6.1730
x_3 (t)	9. 036624	9.036624	9.03662	8.9975	8.1739
x_4 (b)	0.205730	0.205729	0.20573	0.2100	0.2533
$g_1(X)$	-3.6E-9	-1.819E-12	0.0	0.337812	-5758.60377
$g_2(X)$	-2.2E-10	-0.003721	0.0	-353.902604	-255.576901
$g_3(X)$	-6.1 E-14	0.0000000	-5.5511E-17	-0.00120	-0.004400
$g_4(X)$	-3.432984	-3.432983	-3.4329838	-3.411865	-2.982866
$g_5(X)$	-0.235540	-0.235540	-0.2355403	-0.235649	-0.234160
$g_6(X)$	-2.2E-09	0.0000000	-9.0949E-13	-363.232384	-4465.27093
$f(X)$	**1.724852**	**1.724852**	**1.72485084**	**1.74830941**	**2.43311600**

Table 2. Comparison of the solution quality for the pressure vessel problem.

	This paper	Cagnina (2008)	Hu (2003)	Coello (1999)	Deb (1991)
x_1 (T_S)	0.8125	0.8125	0.8125	0.8125	0.9375
x_2 (T_h)	0.4375	0.4375	0.4375	0.4375	0.5000
x_3 (R)	42.098446	42.098445	42.09845	40.3239	48.3290
x_4 (L)	176.636596	176.636595	176.6366	200.0000	112.6790
$g_1(X)$	-2.1E-10	-4.500E-15	0.0	-0.034324	-0.004750
$g_2(X)$	-0.035881	-0.035880	-0.03588	-0.052847	-0.038941
$g_3(X)$	-1.5E-04	-1.164E-10	-0.327	-27.105845	-3,652.876838
$g_4(X)$	-63.363404	-63.363404	-63.3634	-40.0000	-127.321000
$f(X)$	**6,059.714337**	**6,059.714335**	**6,059.131296**	**6,288.7445**	**6410.3811**

Table 3. Comparison of the solution quality for tension/compression spring problem.

	This paper	Cagnina (2008)	Hu (2003)	Coello (1999)
x_1 (d)	0.051644	0.051583	0.051466369	0.051480
x_2 (D)	0.355632	0.354190	0.35138949	0.351661
x_3 (N)	11.35304	11.438675	11.60865920	11.632201
$g_1(X)$	-6.4E-06	-2.000E-16	-0.003336613	-0.002080
$g_2(X)$	-5.3E-06	-1.000E-16	-1.0970128E-04	-0.0001100
$g_3(X)$	-4.0516	-4.048765	-4.0263180998	-4.026318
$g_4(X)$	-0.72848	-0.729483	-0.7312393333	-0.731239
$f(X)$	**0.012665**	**0.012665**	**0.0126661409**	**0.0127047834**

Table 4. Comparison of the solution quality for the speed reducer problem.

	This paper	Cagnina (2008)	Akhtar (2002)
x_1	3.5	3.500000	3.506122
x_2	0.7	0.700000	0.700006
x_3	17	17	17
x_4	7.3	7.300000	7.549126
x_5	7.8	7.800000	7.859330
x_6	3.350215	3.350214	3.365576
x_7	5.286683	5.286683	5.289773
$g_1(X)$	-0.073915	-0.073915	-0.075548
$g_2(X)$	-0.197999	-0.197998	-0.199413
$g_3(X)$	-0.499172	-0.499172	-0.456175
$g_4(X)$	-0.901472	-0.901471	-0.899442
$g_5(X)$	-1.1E-15	0.000000	-0.013213
$g_6(X)$	-6.3E-13	-5.000E-16	-0.001740
$g_7(X)$	-0.7025	-0.702500	-0.702497
$g_8(X)$	-9.9E-15	-1.000E-16	-0.0017388
$g_9(X)$	-0.79583	-0.583333	-0.582608
$g_{10}(X)$	-0.051326	-0.051325	-0.079580
$g_{11}(X)$	-0.010852	-0.010852	-0.017887
$f(X)$	**2,996.3481655**	**2,996.348165**	**3,008.08**

Table 5. Statistical analysis for the solution quality tests.

	This paper			*Coello Coello 1999*		
	Mean Optimal Obj. Funct.	Standard Deviation	Worst Optimal Obj. Funct	*Mean Optimal Obj. Funct.*	*Standard Deviation*	*Worst Optimal Obj. Funct*
Welded beam design	1.7460	0.0446	2.0792	*1.7719*	*0.0112*	*1.7858*
Design of a pressure vessel	0.0127	7.0043E-05	0.0131	*0.0127*	*3.939E-05*	*0.0128*
Tension/compression spring	6.0869E03	39.2722	6.4114E03	*6.293E03*	*7.413*	*6.308E03*
Speed reducer	2.9965E03	0.9121	3.0020E03	*N.A.*	*N.A.*	*N.A.*

Table 6. Comparison of the number of objective function evaluations for engineering functions.

		This paper				*Akhtar 2002*	
	Convergence criterion	Average num. of obj. funct. evals Standard Deviation Min num. of obj. funct. evals Max num. of obj. funct. evals				*Optimal Objective*	*Object. funct. evals*
SSR		on	on	off	off		
PI		on	off	on	off		
Welded beam design	f(X) ≤ 2.4426	940.5 166.9 67 **2,887**	**881.3** 189.9 **55** 4,221	1,385.0 310.8 103 7,962	982.4 **155.8** 112 3,476	*2.4426*	*19,259*
Design of a pressure vessel	f(X) ≤ 6,171	8,444.9 1,420.2 **471** 36,360	**7,802.3** **1,320.1** 619 **25,162**	10,526.9 1,695.6 1,035 34,547	11,721.8 2,271.2 736 65,874	*6,171*	*12,630*
Tension/compression spring	f(X) ≤ 0.0127	6,093.6 **1,138.6** 494 **21,882**	6,874.2 1,294.7 **480** 26,278	**5,653.4** 1,853.3 813 46,229	8,139.1 1,489.2 618 29,170	*N. A.*	*N. A.*
Speed reducer	f(X) ≤ 3,008.08	**5,420.2** **848.9** **377** **19,142**	5,458.2 916.7 484 22,327	10,313.9 1,490.8 540 40,137	11,371.6 1,778.5 605 36,782	*3,008.08*	*19,154*

Table 7. Comparison of the number of objective function evaluations for mathematical functions.

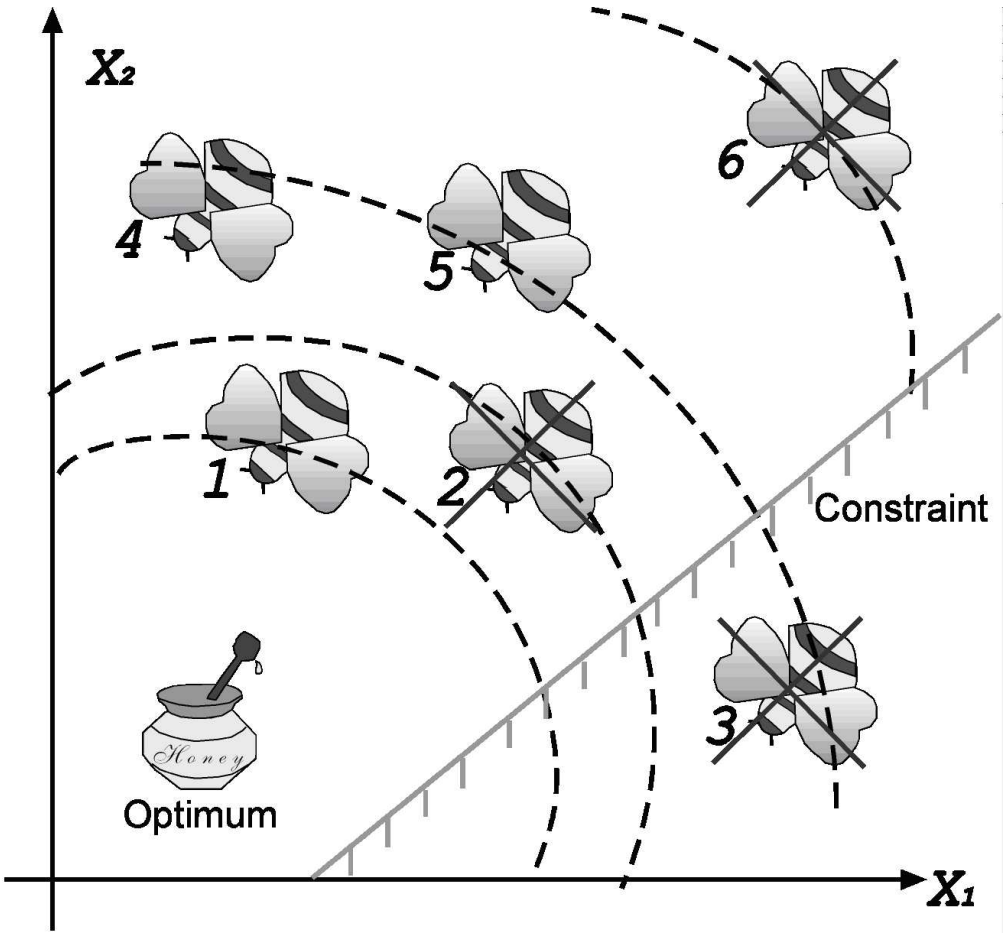
	Convergence criterion	Average num. of obj. funct. evals Standard Deviation Min num. of obj. funct. evals Max num. of obj. funct. evals			
SSR		on	on	off	off
PI		on	off	on	off
Ackley (dimension = 30)	f(X) ≤ 1E-3	**238,530** 16,678 110,342 **582,564**	246,060 **16,279** **76,954** 597,749	455,000 37,862 169,164 1,152,523	387,470 27,614 206,462 1,104,774
Griewangk (dimension = 30)	f(X) ≤ 1E-3	1,114,200 174,020 174,283 5,930,814	**609,670** **63,053** **122,089** **2,058,378**	1,997,500 255,300 231,115 8,742,315	1,928,600 159,560 183,459 5,847,714
Alpine (dimension = 30)	f(X) ≤ 1E-3	1,731,300 150,230 223,797 3,932,548	**933,090** **63,843** **179,321** **1,986,622**	2,586,000 200,510 350,155 6,740,704	2,770,600 182,230 781,835 5,710,433
Sphere (dimension = 30)	f(X) ≤ 1E-3	325,370 21,671 **79,157** **652,663**	**273,850** **21,310** 139,150 829,375	731,390 49,548 252,385 1,676,363	550,120 26,580 293,093 972,296

Table 8. The effect of PSO parameters on engineering functions solution costs. Swarm size = 20.

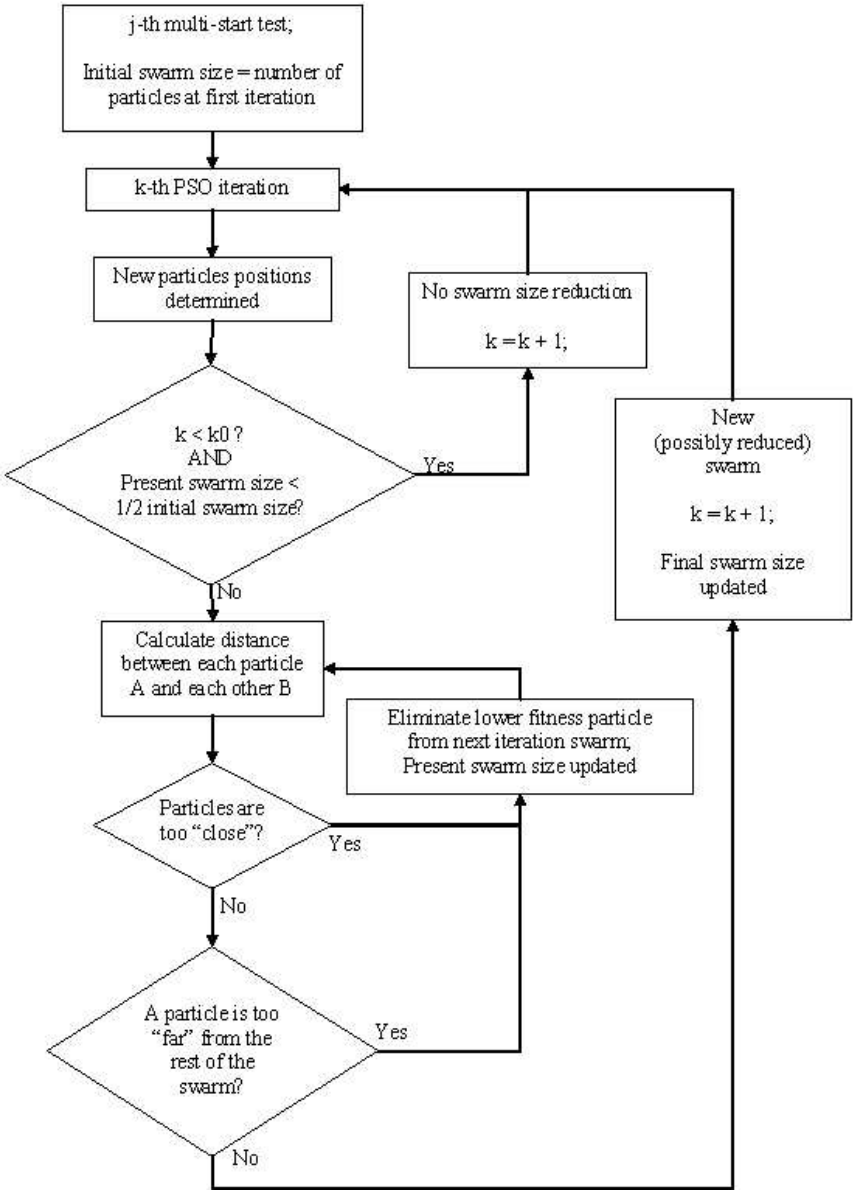
	Convergence criterion		Average number of objective function evaluations Standard deviation			
SSR		$c_1=c_2$	on	on	off	off
PI			on	off	on	off
Welded beam design	f(X) ≤ 2.4426	1.00	2,166.3 811.7	**1,083.8** **307.4**	1,569.0 598.6	2,061.8 775.2
		1.49445	**798.2** **88.6**	983.3 100.3	1,209.6 175.8	976.7 135.6
		2.00	673.7 **115.8**	**430.8** 153.6	658.8 277.3	850.6 208.5
Design press. vessel	f(X) ≤ 6,171	1.00	10,999.7 3,045.1	**5,833.1** **1,598.1**	13,233.3 3,210.4	12,306.1 3,484.0
		1.49445	**7,998.6** 816.2	9,623.8 941.0	9,016.0 **788.8**	10,088.5 1,069.9
		2.00	37,259.7 8,010.6	**25,987.1** **5,998.2**	38,395.9 7,140.0	54,672.4 17,839.9
Tens./compr. spring	f(X) ≤ 0.0127	1.00	**5,269.2** **1,170.7**	7,381.2 1,827.3	9,865.1 3,628.0	6,739.9 1,536.4
		1.49445	8,145.9 724.7	**5,815.0** **464.4**	8,032.7 723.1	6,817.0 593.1
		2.00	24,135 5,297.2	29,995.6 8,659.4	**21,250.3** **5,164.2**	24,909.5 8,013.8
Speed reducer	f(X) ≤ 3,008.08	1.00	96,735.1 32,574.8	**83,793.7** **21,713.3**	245,156.3 61,231.3	428,204.5 86,650.1
		1.49445	**4,427.1** 500.2	5,023.5 **407.3**	8,671.4 815.5	11,169.6 1,084.4
		2.00	4,152.3 1,145.6	**3,085.2** **806.3**	10,522.7 2,584.1	8,916.5 2,908.5

Table 9. The effect of swarm size on engineering functions solution costs (c_1=c_2=1.49445).

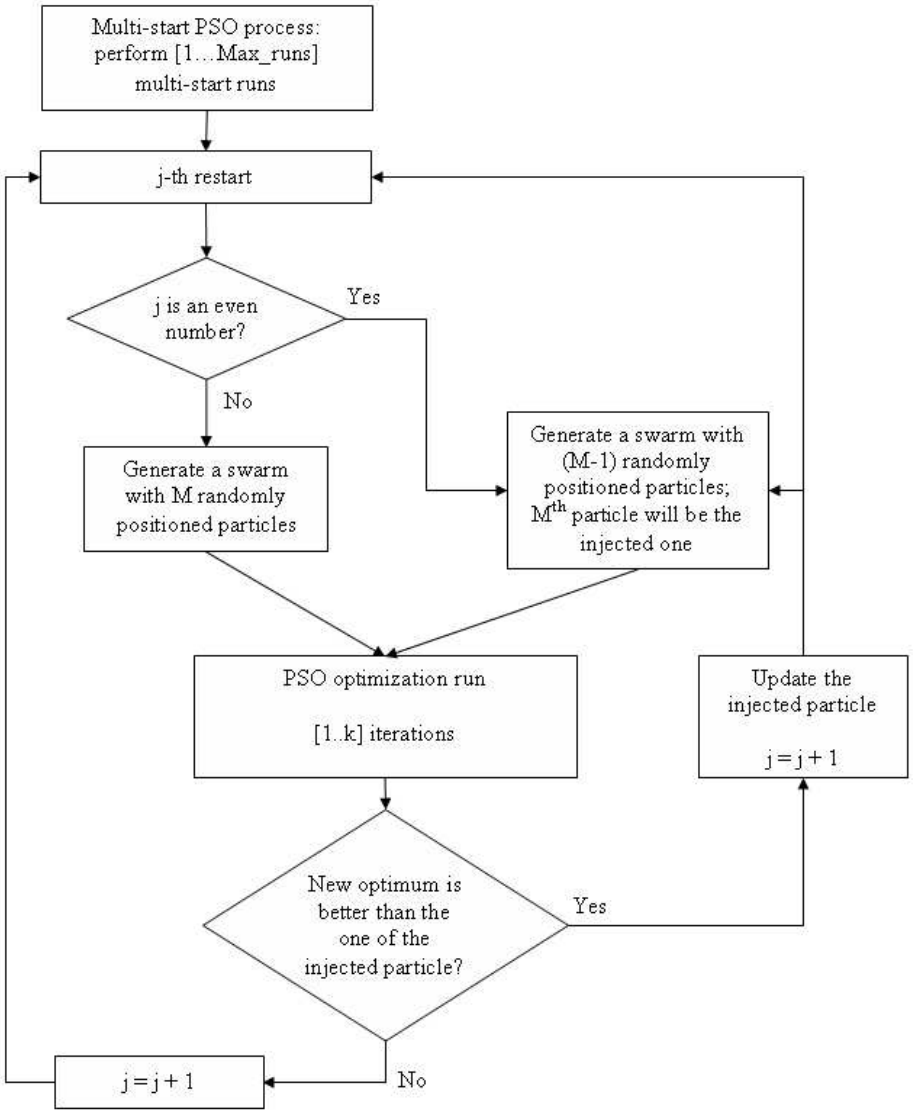
	Convergence criterion		Average number of objective function evaluations Standard deviation			
SSR		Swarm size	on	on	off	off
PI			on	off	on	off
Welded beam design	f(X) ≤ 2.4426	20	**798.2** **88.6**	983.3 100.3	1,209.6 175.8	976.7 135.6
		50	**646.2** 79.7	691.9 **74.6**	964.9 135.9	853.4 125.5
		100	**607.9** **45.9**	751.3 86.3	953.3 155.0	785.6 114.1
Design press. vessel	f(X) ≤ 6,171	20	**7,998.6** 816.2	9,623.8 941.0	9,016.0 **788.8**	10,088.5 1,069.9
		50	10,897.3 941.8	**7,678.6** **604.8**	11,958.6 1,069.5	18,162.5 1,743.5
		100	**15,141.4** 1,535.4	16,627.1 **1,490.0**	20,544.0 2,930.0	23,881.1 2,297.0
Tens./compr. spring	f(X) ≤ 0.0127	20	8,145.9 724.7	**5,815.0** **464.4**	8,032.7 723.1	6,817.0 593.1
		50	**6,890.2** **528.0**	9,652.4 843.7	11,217.7 1,129.6	10,324.0 959.0
		100	14,420.9 **1,330.0**	**14,304.2** 1,334.4	16,905.3 1,595.2	20,770.6 2,172.4
Speed reducer	f(X) ≤ 3,008.08	20	**4,427.1** 500.2	5,023.5 **407.3**	8,671.4 815.5	11,169.6 1,084.4
		50	**4,343.7** **359.3**	4,836.7 407.2	4,897.0 428.7	4,780.8 446.2
		100	**4,303.7** **434.8**	5,828.3 693.8	4,804.3 449.7	5,733.2 489.8



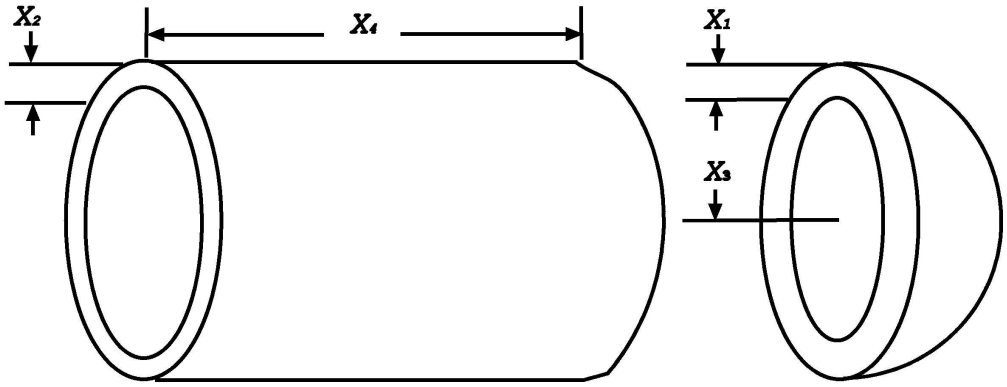
Schematic of the selection mechanism among particles (SSR).
346x320mm (96 x 96 DPI)



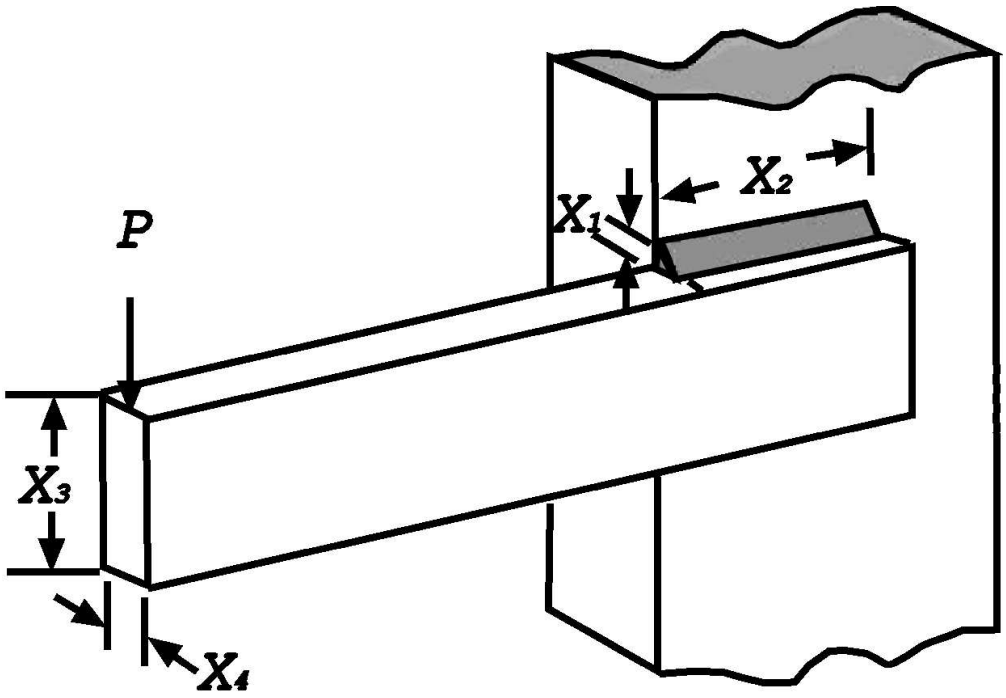
Schematic diagram of the SSR technique.
150x204mm (96 x 96 DPI)



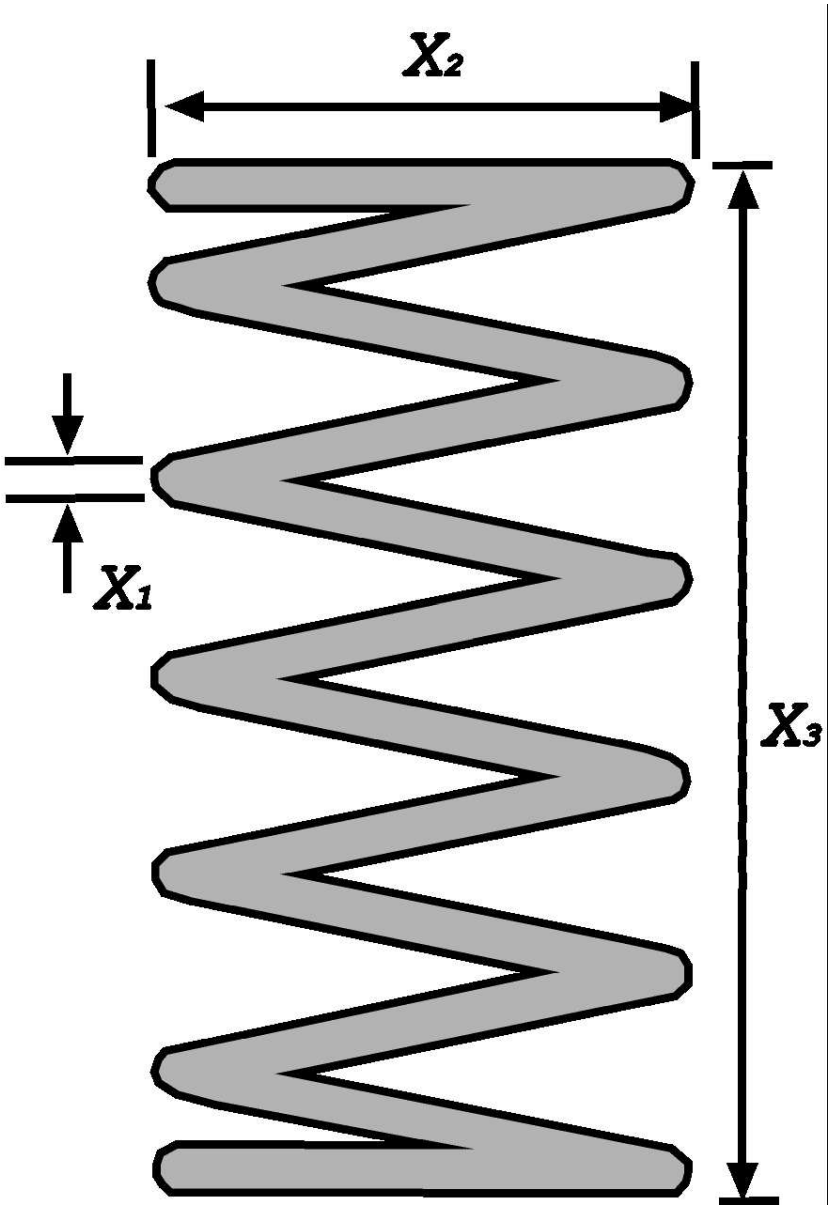
Schematic diagram of the PI technique.
182x206mm (96 x 96 DPI)



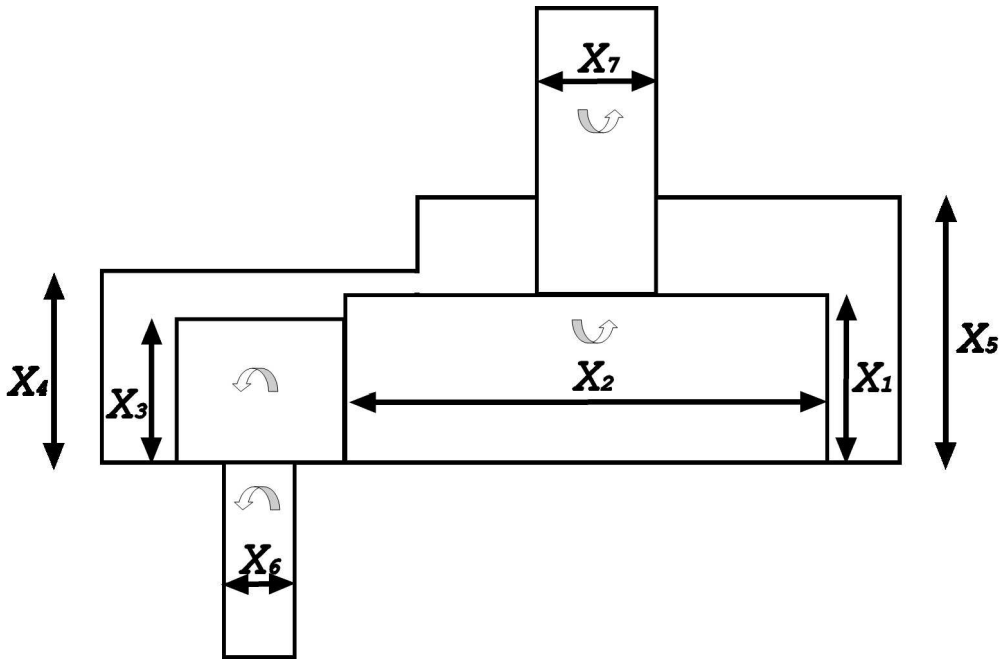
The pressure vessel design problem.
199x77mm (300 x 300 DPI)



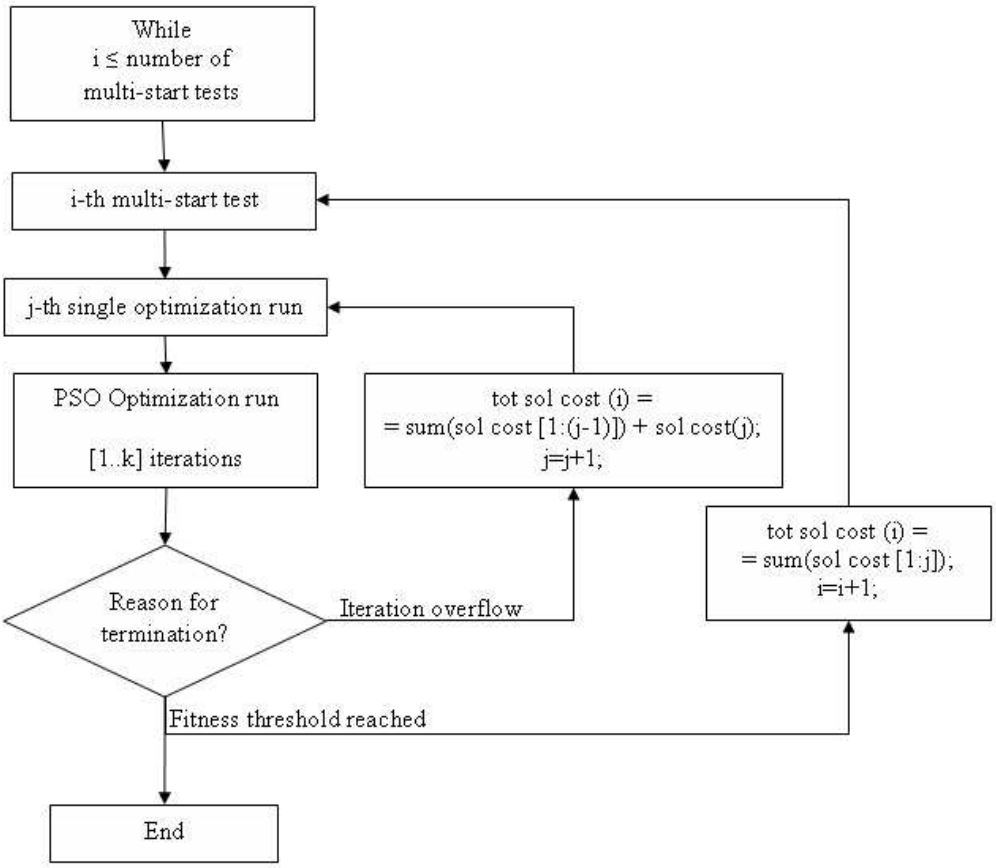
The welded beam design problem.
104x73mm (300 x 300 DPI)



The tension/compression spring design problem.
74x108mm (300 x 300 DPI)



The speed reducer design problem.
412x269mm (96 x 96 DPI)



Schematic diagram of the numerical procedure for solution cost tests.
167x147mm (96 x 96 DPI)