AIAA 2002–1235

Particle Swarm Optimization

Gerhard Venter (gventer@vrand.com)
Vanderplaats Research and Development, Inc.
1767 S 8th Street, Suite 100, Colorado Springs, CO 80906

Jaroslaw Sobieszczanski-Sobieski (j.sobieski@larc.nasa.gov)
NASA Langley Research Center
MS 240, Hampton, VA 23681-2199

43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference
April 22-25, 2002
Denver, Colorado

AIAA 2002–1235

Particle Swarm Optimization

Gerhard Venter (gventer@vrand.com) *
Vanderplaats Research and Development, Inc.
1767 S 8th Street, Suite 100, Colorado Springs, CO 80906

Jaroslaw Sobieszczanski-Sobieski (j.sobieski@larc.nasa.gov) †
NASA Langley Research Center
MS 240, Hampton, VA 23681-2199

The purpose of this paper is to show how the search algorithm known as particle swarm optimization performs. Here, particle swarm optimization is applied to structural design problems, but the method has a much wider range of possible applications. The paper's new contributions are improvements to the particle swarm optimization algorithm and conclusions and recommendations as to the utility of the algorithm. Results of numerical experiments for both continuous and discrete applications are presented in the paper. The results indicate that the particle swarm optimization algorithm does locate the constrained minimum design in continuous applications with very good precision, albeit at a much higher computational cost than that of a typical gradient based optimizer. However, the true potential of particle swarm optimization is primarily in applications with discrete and/or discontinuous functions and variables. Additionally, particle swarm optimization has the potential of efficient computation with very large numbers of concurrently operating processors.

Introduction

MOST general-purpose optimization software used in industrial applications makes use of gradient-based algorithms, mainly due to their computational efficiency. However, in recent years non-gradient based, probabilistic search algorithms have attracted much attention from the research community. These algorithms generally mimic some natural phenomena, for example genetic algorithms and simulated annealing. Genetic algorithms model the evolution of a species, based on Darwin's principle of survival of the fittest,[1] while simulated annealing is based on statistical mechanics and models the equilibrium of large numbers of atoms during an annealing process.[2]

Although these probabilistic search algorithms generally require many more function evaluations to find an optimum solution, as compared to gradient-based algorithms, they do provide several advantages. These algorithms are generally easy to program, can efficiently make use of large numbers of processors, do not require continuity in the problem definition, and generally are better suited for finding a global, or near global, solution. In particular these algorithms are ideally suited for solving discrete and/or combinatorial type optimization problems.

In this paper, a fairly recent type of probabilistic search algorithm, called Particle Swarm Optimization (PSO), is investigated. The PSO algorithm is based on a simplified social model that is closely tied to swarming theory. The algorithm was first introduced by Kennedy and Eberhart.[3,4] A physical analogy might be a swarm of bees searching for a food source. In this analogy, each bee (referred to as a particle here) makes use of its own memory as well as knowledge gained by the swarm as a whole to find the best available food source.

Since it was originally introduced, the PSO algorithm has been studied by a number of different authors.[5–8] These authors concentrated mostly on multi-modal mathematical problems that are important in the initial research of any optimization algorithm, but are of little practical interest. Few ap-

*Senior R&D Engineer, AIAA Member

†Senior Research Scientist, Analytical and Computational Methods Branch, Structures and Materials Competency, AIAA Fellow

plications of the algorithm to structural and multidisciplinary optimization are known. Two examples are by Fourie and Groenwold who considered an application to shape and size optimization[9] and an application to topology optimization.[10]

The present paper focuses on enhancements to the basic PSO algorithm. These include the introduction of a convergence criterion and dealing with constrained and discrete problems. The enhanced version of the algorithm is applied to the design of a ten design variable cantilevered beam. Both continuous and integer/discrete versions of the problem are studied.

Basic Particle Swarm Optimization Algorithm

Particle swarm optimization makes use of a velocity vector to update the current position of each particle in the swarm. The position of each particle is updated based on the social behavior that a population of individuals, the swarm in the case of PSO, adapts to its environment by returning to promising regions that were previously discovered.[5] The process is stochastic in nature and makes use of the memory of each particle as well as the knowledge gained by the swarm as a whole. The outline of a basic PSO algorithm is as follows:

1. Start with an initial set of particles, typically randomly distributed throughout the design space
2. Calculate a velocity vector for each particle in the swarm
3. Update the position of each particle, using its previous position and the updated velocity vector
4. Go to Step 2 and repeat until convergence

The scheme for updating the position of each particle is shown in (1)

$$\boldsymbol{x}_{k+1}^{i} = \boldsymbol{x}_{k}^{i} + \boldsymbol{v}_{k+1}^{i}\Delta t \tag{1}$$

where $\boldsymbol{x}_{k+1}^{i}$ represents the position of particle i at iteration $k+1$ and $\boldsymbol{v}_{k+1}^{i}$ represents the corresponding velocity vector. A unit time step (Δt) is used throughout the present work.

The scheme for updating the velocity vector of each particle depends on the particular PSO algorithm under consideration. A commonly used scheme was introduced by Shi and Eberhart,[6] as shown in (2)

$$\boldsymbol{v}_{k+1}^{i} = w\boldsymbol{v}_{k}^{i} + c_1 r_1 \frac{\left(\boldsymbol{p}^{i} - \boldsymbol{x}_{k}^{i}\right)}{\Delta t} + c_2 r_2 \frac{\left(\boldsymbol{p}_{k}^{g} - \boldsymbol{x}_{k}^{i}\right)}{\Delta t} \tag{2}$$

where r_1 and r_2 are random numbers between 0 and 1, $\boldsymbol{p}^{i}$ is the best position found by particle i so far and $\boldsymbol{p}_{k}^{g}$ is the best position in the swarm at time k. Again, a unit time step (Δt) is used throughout the present work. There are three problem dependent parameters, the inertia of the particle (w), and two "trust" parameters c_1 and c_2. The inertia controls the exploration properties of the algorithm, with larger values facilitating a more global behavior and smaller values facilitating a more local behavior. The trust parameters indicate how much confidence the current particle has in itself (c_1) and how much confidence it has in the swarm (c_2).

Fourie and Groenwold[9] proposed a slight modification to (2) for their structural design applications. They proposed using the best position in the swarm to date $\boldsymbol{p}^{g}$, instead of the best position in the swarm at iteration k, $\boldsymbol{p}_{k}^{g}$. Both approaches were investigated and it was found that (2) works slightly better for our applications. As a result (2) is used throughout the present work.

Initial Swarm

The initial swarm is generally created such that the particles are randomly distributed throughout the design space, each with a random initial velocity vector. In the present work, (3) and (4) are made use of to obtain the random initial position and velocity vectors.

$$\boldsymbol{x}_{0}^{i} = \boldsymbol{x}_{min} + r_1\left(\boldsymbol{x}_{max} - \boldsymbol{x}_{min}\right) \tag{3}$$

$$\boldsymbol{v}_{0}^{i} = \frac{\boldsymbol{x}_{min} + r_2\left(\boldsymbol{x}_{max} - \boldsymbol{x}_{min}\right)}{\Delta t} \tag{4}$$

In (3) and (4), r_1 and r_2 are random numbers between 0 and 1, $\boldsymbol{x}_{min}$ is the vector of lower bounds and $\boldsymbol{x}_{max}$ is the vector of upper bounds for the design variables.

The influence of the initial swarm distribution on the effectiveness of the PSO algorithm is studied by considering the initial particle distribution. Instead of using a random distribution, a space filling design of experiments (DOE) was used to distribute the initial swarm in the design space. However, these numerical experiments indicate that the initial distribution of the particles (random as compared to using the space filling DOE) is not important to the overall performance of the PSO algorithm. The reason being that the swarm changes dynamically throughout the optimization process until an optimum solution is reached. As a result, the precise distribution of the initial swarm is not important, as long as it is fairly well distributed throughout the design space. In the present work, all initial swarms are randomly distributed.

Problem Parameters

The basic PSO algorithm has three problem dependent parameters, w, c_1 and c_2. The literature[3] proposes using $c_1 = c_2 = 2$, so that the mean of the stochastic multipliers of (2) is equal to 1. Additionally, Shi and Eberhart[7] suggest using $0.8 < w < 1.4$, starting with larger w values (a more global search behavior) that is dynamically reduced (a more local search behavior) during the optimization.

The present work showed that having each particle put slightly more trust in the swarm (larger c_2 value) and slightly less trust in itself (smaller c_1 value) seems to work better for the structural design problems considered here. Using $c_1 = 1.5$ and $c_2 = 2.5$ works well in all example problems considered. Additionally, dynamically adjusting the w value has several advantages. First, it results in faster convergence to the optimum solution and second, it makes the w parameter problem independent. The scheme that dynamically adjusts the w value is discussed in more detail in the next section.

Enhancements to the Basic Algorithm

The basic PSO algorithm summarized in (1) and (2) has been used in the literature to demonstrate the PSO algorithm on a number of test problems. However, the goal here is to develop an implementation of the basic algorithm that would be more general in nature, applicable to a wide range of design problems. To achieve this goal, a number of enhancements to the basic algorithm were investigated. These enhancements are discussed in more detail in this section.

Convergence Criterion

A robust convergence criterion is important for any general-purpose optimizer. Most implementations of the PSO algorithm also implement some convergence criterion. A convergence criterion is necessary to avoid any additional function evaluations after an optimum solution is found. Ideally, the convergence criterion should not have any problem specific parameters. The convergence criterion used here is very basic. The maximum change in the objective function is monitored for a specified number of consecutive design iterations. If the maximum change in the objective function is less than a predefined allowable change, convergence is assumed.

Problem Parameters

The $c_1 = 1.5$ and $c_2 = 2.5$ trust parameter values discussed earlier are used throughout the present work. These parameters seem to be fairly problem independent, but further study is required.

The inertia weight parameter w is adjusted dynamically during the optimization, as suggested by Shi and Eberhart,[7] with an example implementation by Fourie and Groenwold.[9] Shi and Eberhart[7] proposed linearly decreasing w during the first part of the optimization, while Fourie and Groenwold[9] decreased the w value with a fraction if no improvement has been made for a predefined number of consecutive design iterations.

In the present paper a different implementation is proposed, based on the coefficient of variation (COV) of the objective function values. The goal is to change the w value in a problem independent way, with no interaction from the designer. A starting value of $w = 1.4$ is used to initially accommodate a more global search and is dynamically reduced to no less than $w = 0.35$. The idea is to terminate the PSO algorithm with a more local search. The w value is adjusted using (5)

$$w_{new} = w_{old} f_w \tag{5}$$

where w_{new} is the newly adjusted w value, w_{old} is the previous w value and f_w is a constant between 0 and 1. Smaller f_w values would result in a more dramatic reduction in w, that would in turn result in a more local search. In the present work $f_w = 0.975$ is used throughout, resulting in a PSO algorithm with a fairly global search characteristic.

The w value is not adjusted at each design iteration. Instead the coefficient of variation (COV) of the objective function values for a subset of best particles is monitored. If the COV falls below a specified threshold value, it is assumed that the algorithm is converging towards an optimum solution and (5) is applied. A general equation to calculate the COV for a set of points is provided in (6)

$$COV = \frac{StdDev}{Mean} \tag{6}$$

where $StdDev$ is the standard deviation and $Mean$ is the mean value for the set of points. In the present work, a subset of the best 20% of particles from the swarm are monitored and a COV threshold of 1.0 is used.

Constrained Optimization

The basic PSO algorithm is defined for unconstrained problems only. Since most engineering problems are constrained in one way or the other, it is important to add the capability of dealing with constrained optimization problems. It was decided to deal with constraints by making use of a quadratic exterior penalty function. This technique is often used to deal with constrained problems in genetic

algorithms. In the current implementation, the objective function is penalized as shown in (7) when one or more of the constraints are violated.

$$\tilde{f}(\boldsymbol{x}) = f(\boldsymbol{x}) + \alpha \sum_{i=1}^{m} \max\left[0, g_i(\boldsymbol{x})\right]^2 \qquad (7)$$

In (7), $f(\boldsymbol{x})$ is the original objective function, α is a large penalty parameter, $g_i(\boldsymbol{x})$ is the set of all constraints (with violated constraints having values larger than zero), and $\tilde{f}(\boldsymbol{x})$ is the new, penalized, objective function. In the present work a penalty parameter of $\alpha = 10^8$ is used.

Particles With Violated Constraints

When dealing with constrained optimization problems, special attention to particles with violated constraints needs to be paid. This issue was not addressed in the literature, thus a new enhancement to the basic PSO algorithm is proposed here.

It is preferable to restrict the velocity vector of a violated particle to a usable, feasible direction e.g., Vanderplaats,[11] a direction that would reduce the objective function while pointing back to the feasible region of the design space. Unfortunately, this would require gradient information for each violated particle. Currently, one of the attractive features of the PSO algorithm is that no gradient information is required and thus the concept of calculating a usable, feasible direction is not viable.

Instead, a simple modification to (2) for particles with one or more violated constraints is proposed. The modification can be explained by considering particle i, which is assumed to have one or more violated constraints at iteration k. By re-setting the velocity vector of particle i at iteration k to zero, the velocity vector at iteration $k+1$ is obtained as

$$\boldsymbol{v}_{k+1}^i = c_1 r_1 \frac{\left(\boldsymbol{p}^i - \boldsymbol{x}_k^i\right)}{\Delta t} + c_2 r_2 \frac{\left(\boldsymbol{p}_k^g - \boldsymbol{x}_k^i\right)}{\Delta t} \qquad (8)$$

The velocity of particle i at iteration $k+1$ is thus only influenced by the best point found so far for the particle itself and the current best point in the swarm. In most cases this new velocity vector will point back to a feasible region of the design space. The result is to have the violated particle move back towards the feasible design space in the next design iteration.

Discrete/Integer Design Variables

Unlike a genetic algorithm that is inherently a discrete algorithm, the PSO algorithm is inherently a continuous algorithm. Although the PSO algorithm is able to find optimum solutions to continuous problems very accurately, the associated computation cost is high compared to gradient-based algorithms. However, the true potential of the PSO algorithm is in applications with discrete and/or discontinuous functions and variables. In other words, the algorithm is expected to excel in applications where a gradient-based algorithm is not a viable alternative.

In this paper, two different modifications to the basic PSO algorithm that allow the solution of problems with discrete variables are considered. The first approach is straight forward. The position of each particle is modified to represent a discrete point, by rounding each position coordinate to its closest discrete value after applying (1).

The second approach is more elaborate. The position of each particle is modified to represent a discrete point, by considering a set of candidate discrete values about the new continuous point, obtained after applying (2). The candidate discrete points are obtained by rounding each continuous position coordinate to its closest upper and lower discrete values. For a problem with $nDvar$ discrete design variables, this process will result in $nDvar^2$ candidate discrete points. For a two dimensional, integer problem, this would produce four candidate discrete points, located at the vertices of a two dimensional rectangle. The discrete point to use as the new position for the particle is selected from the candidate set of discrete points as the point with the shortest perpendicular distance to the velocity vector. There are several enhancements to this scheme, based on the direction of the velocity vector, that will substantially reduce the number of candidate discrete points, but a more detail discussion is beyond the scope of this paper. Reducing the number of candidate discrete points is especially important for problems with larger numbers of design variables.

The more elaborate approach is expected to result in a more efficient integer/discrete algorithm. In contrast to the expectation, numerical experiments indicate that there is no significant difference in the performance of the PSO algorithm when using the first as compared to the second approach. Since there is no advantage to using the more elaborate approach, this approach was discarded and the simpler rounding approach is used for all integer/discrete problems presented in this paper.

Additional Randomness

To avoid premature convergence of the algorithm, the literature mentions the possible use of a craziness operator[3] that adds randomness to the swarm. The craziness operator acts similarly to the mutation operator in genetic algorithms. However, there does not seem to be concensus in the literature whether

or not the craziness operator should be applied. After introducing the craziness operator, Kennedy and Eberhart[3] conclude that this operator may not be necessary, while Fourie and Groenwold[9] reintroduced the craziness operator for their structural design problems. It was thus decided to implement a craziness operator here and test it's effectiveness for our structural design problems.

The originally proposed craziness operator identifies a small portion of randomly selected particles at each iteration for which the velocity vector is randomly changed. In the present work, the craziness operator is modified. The craziness operator used here also identifies a small number of particles at each design iteration, but instead of changing the velocity vector, both the position and the velocity vector are changed. The position of the particles are changed randomly, while the velocity vector of each modified particle is reset to only the second component of (2) as shown in (9).

$$\boldsymbol{v}_{k+1}^{i} = c_1 r_1 \frac{\left(\boldsymbol{p}^{i} - \boldsymbol{x}_{k}^{i}\right)}{\Delta t} \tag{9}$$

In the present implementation, the particles to modify are identified using the COV for the objective function values of all particles, at the end of each design iteration. If the COV falls below a predefined threshold value, it is assumed that the swarm is becoming too uniform. In this case, particles that are located far from the center of the swarm are identified, using the standard deviation of the position coordinates of the particles. Particles that are located more than 2 standard deviations from the center of the swarm are subjected to the craziness operator. In the present work, a COV threshold value of 0.1 is used.

Example Problem

To study the behavior of the PSO algorithm, a cantilevered beam example, simular to that considered by Vanderplaats[11] was chosen. A schematic representation of the example problem, including material properties, are shown in Fig. 1.

The beam is modeled using five segments of equal length and the design problem is defined as minimizing the material volume of the beam, subject to maximum bending stress constraints for each segment. The design variables are the height (h) and width (b) of each segment, resulting in ten independent design variables. Two cases are considered. The first is a continuous design problem where the height of each segment is allowed to vary between 50 cm and 100 cm, while the width is allowed to vary between 0.5 cm and 10 cm. The second is an integer/discrete case where all ten design variables are restricted to integer values only. For the second case, the height of each segment is allowed to vary between 50 cm and 100 cm, while the width is allowed to vary between 1 cm and 10 cm.

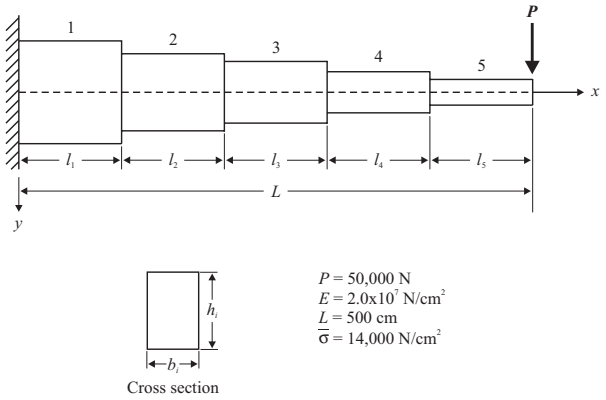

Fig. 1 Cantilevered beam example problem

The bending stress is obtained from the well known bending stress equation shown in (10)

$$\sigma = \frac{My}{I} \tag{10}$$

where σ represents the bending stress, M the applied bending moment, I the moment of inertia and y is the vertical distance, measured from the neutral axis, where the stress is calculated. For this problem $M = P(L - x)$ and $I = \frac{1}{12}bh^3$, while the maximum stress value occurs at $y = h/2$. The height thus has a much larger influence on the bending stress, as compared to the width of the beam. It is reasonable to assume that the optimizer would minimize the weight by keeping the width constant and equal to its lower bound, while changing the height of the beam. In this case, the theoretical solution for the height is

$$h = \sqrt{\frac{6P(L-x)}{\overline{\sigma}b}} \tag{11}$$

where P is the applied tip load, x is the horizontal distance measured from the root of the beam and $\overline{\sigma}$ is the allowable stress limit.

To verify the above assumptions, and to study how well a gradient-based optimizer would solve this problem, it was decided to model and solve the continuous case using the GENESIS[12] structural analysis and optimization code. The theoretical optimum results for a beam with uniform height and width across the span, as well as the GENESIS and theoretical optimum results for a beam with five segments are summarized in Table 1. The case with uniform height and width was obtained by setting

the width equal to its lower bound and calculating the height from (11) to have the maximum stress at the root equal to the allowable stress. The case with constant height and width may be considered as a baseline from which the optimizer makes improvement.

Table 1 Comparison between continuous GENESIS and theoretical results

Parameter	Baseline	GENESIS	Theory
Volume (cm^3)	36683	27498	27438
b_1 (cm)	0.5	0.5	0.5
b_2 (cm)	0.5	0.5	0.5
b_3 (cm)	0.5	0.5	0.5
b_4 (cm)	0.5	0.5	0.5
b_5 (cm)	0.5	0.5	0.5
h_1 (cm)	146.73	146.73	146.39
h_2 (cm)	146.73	131.16	130.93
h_3 (cm)	146.73	113.16	113.39
h_4 (cm)	146.73	92.78	92.58
h_5 (cm)	146.73	65.61	65.47

Table 1 verifies both the assumptions that lead to (11) and the GENESIS results. GENESIS solved the problem using 12 finite element analyses. However, it should be noted that GENESIS makes use of advanced approximation techniques to reduce the required function evaluations for solving structural optimization problems. A general purpose gradient-based optimizer would most probably require between 100 and 300 function evaluations to solve this problem.

Finding the theoretical optimum for the integer/discrete case is a more daunting task. Instead, we'll determine tight upper and lower bounds for the objective function value. A lower bound may be obtained from (11), using the lower bound values of 1.0 for all b_i, thus producing an optimum answer with half the variables being integer/discrete and the other half being continuous. An upper bound may be obtained by considering three discrete points, obtained by rounding the continuous h_i values obtained from (11). The three points are obtained by rounding all h_i values up, rounding all h_i values down and rounding all h_i values to their closest integer value. It turns out that only the case where all h_i values are rounded up produce a feasible design, thus providing an upper bound for the integer/discrete optimum solution.

The theoretical results for a beam with uniform height and width across the span, using the root dimensions of the lower bound solution, and the upper and lower bounds for the integer/discrete design problem are shown in Table 2. The difference in the objective function values of the calculated upper and lower bounds is less than 1% and Table 2 thus provides tight upper and lower bounds for the integer/discrete solution. Again, the beam with constant height and width may be considered as a baseline from which the optimization makes improvement.

Table 2 Upper and lower bound solutions for the integer/discrete case

Parameter	Baseline	Lower Bound	Upper Bound
Volume (cm^3)	51755	38803	39100
b_1 (cm)	1.0	1.0	1.0
b_2 (cm)	1.0	1.0	1.0
b_3 (cm)	1.0	1.0	1.0
b_4 (cm)	1.0	1.0	1.0
b_5 (cm)	1.0	1.0	1.0
h_1 (cm)	103.51	103.51	104
h_2 (cm)	103.51	92.58	93
h_3 (cm)	103.51	80.18	81
h_4 (cm)	103.51	65.47	66
h_5 (cm)	103.51	46.29	47

Results

The PSO algorithm was used to analyze the beam example problem, employing the elementary strength of materials approach described in the previous section. As discussed, two design problems were considered, the first is a continuous problem and the second an integer/discrete problem. For each design problem, the influence of two enhancements to the basic algorithm is considered. The first is the proposed craziness operator. As mentioned previously, there seems to be disagreement in the literature as to the usefulness of the craziness operator. The second is resetting the velocity vectors of violated design points. Resetting the velocity vectors is a new feature that has not been studied previously. To fully investigate the influence of these two enhancements, all possible combinations of using and not using the enhancements were considered, resulting in four possible combinations. The PSO algorithm was first run for a fixed number of function evaluations and then using the proposed convergence criterion. In all cases, a swarm size of 300 particles was used.

Each run was repeated 50 times and the best, worst, mean and standard deviation of the best objective function from each of the 50 repetitions were recorded. For the runs where the convergence criterion was used, the best, worst, mean and standard

deviation of the number of function evaluations to convergence for each of the 50 repetitions were also recorded. For all runs the same PSO parameters were used, as summarized in Table 3. The w, c_1 and c_2 values were determined as discussed in previous sections of this paper. The number of particles (swarm size) was selected as a tradeoff between cost and reliability. Smaller swarm sizes required less function evaluations for convergence, but decreased the reliability of the algorithm. Larger swarm sizes required more function evaluations for convergence, but increased the reliability of the algorithm.

Table 3 PSO parameters used in example problems

Parameter	Value
Number of particles	300
Initial inertia weight, w	1.4
Trust parameter 1, c_1	1.50
Trust parameter 2, c_2	2.50

Each run is identified by the combination of enhancements used during that run, as summarized in Table 4. From Table 4, `R` would represent resetting the velocities of violated particles only, while `CR` would represent using the craziness operator and resetting the velocities of violated particles.

Table 4 PSO enhancement summary

Option	Definition
`C`	Apply craziness operator
`R`	Reset velocities of violated particles

Fixed Number of Function Evaluations

First, each of the four combinations was evaluated using a fixed number of design iterations equal to 50, resulting in a total number of function evaluations equal to 15000. The statistical results obtained from 50 repetitions for each combination are summarized in Table 5 for the continuous design problem and in Table 6 for the integer/discrete design problem.

Table 5 Objective function (material volume) statistics for the continuous design problem

Option	Mean	StdDev	Best	Worst
—	41383	18548	27610	95547
`R`	31897	12247	27438	91809
`C`	43232	19866	30150	92208
`CR`	33534	15394	27439	110824

Table 6 Objective function (material volume) statistics for the integer design problem

Option	Mean	StdDev	Best	Worst
—	67380	20707	39900	112196
`R`	42822	10153	39100	89491
`C`	62562	17690	39100	107596
`CR`	42253	10234	39100	85086

By comparing the statistical data, especially the mean and standard deviation values, from Tables 5 and 6 it is clear that resetting the velocity vectors of the violated design points has a significant and positive influence on the performance of the PSO algorithm. In contrast, the craziness operator does not appear to have a big influence. It is not clear if combining the craziness operator and resetting the velocity vectors of the violated design points results in any additional improvements over just resetting the velocity vectors without the craziness operator. Finally, the standard deviation clearly shows that the algorithm is more successful in solving the discrete problem (Table 6) than the continuous problem (Table 5). This was expected, since the discrete problem results in a smaller design space as compared to the continuous problem.

For the case with a fixed number of function evaluations, it is possible to compare the optimum results obtained by the PSO algorithm against a random search, using the same number of function valuations. We performed a random search with 15000 analyses, again repeating the process 50 times, and recorded the statistics for the best objective function from each repetition. The results are summarized in Table 7.

Table 7 Objective function (material volume) statistics for the continuous problem using a random search

Mean	StdDev	Best	Worst
176956	28349	117115	261619

When comparing the results from Table 7 with that from Table 5, it is clear that the random search has a terrible performance as compared to the PSO algorithm, using the same number of function evaluations.

Convergence Criterion

Next the runs of Tables 5 and 6 are repeated, using the proposed convergence criterion. For convergence the objective function is required not to change more than 0.1% in 10 consecutive design

iterations. A maximum of 500 design iterations, equivalent to 150000 analyses, was allowed for cases where the algorithm did not converge.

The statistical results for both the cost (number of function evaluations) and the objective function values obtained from 50 repetitions are summarized in Table 8 for the continuous design problem. Table 9 contains the corresponding results for the integer/discrete design problem.

The results are simular to that of Tables 5 and 6. Resetting the velocity vectors of violated particles has a big influence on the performance of the PSO algorithm, while applying the craziness operator has a much smaller influence. It would appear that combining the two enhancements does have a positive influence in that the number of function evaluations are reduced while better results (smaller mean and standard deviation for the objective function values) are obtained. Again, the discrete problem is solved more efficiently than the continuous problem.

Note that 300 particles were used in all cases. If one considers the mean cost when using both enhancements, an average of 49.24 design iterations is required to solve the continuous problem and 32.62 to solve the integer/discrete problem. Within a design iteration, all analyses are independent of each other and can be easily parallelized with minimal inter-process communication. It is thus reasonable to expect near perfect speedup within a design iteration, when adding more processors. In the extreme case, using 300 processors should allow the designer to solve the ten design variable continuous problem in the equivalent time of 49 analyses, while the integer/discrete case can be solved in the equivalent time of only 32 analyses. If more processors are available, the number of particles considered can be increased to the number of available processors. Using more particles has the advantage of increasing the robustness of the algorithm and reducing the number of design iterations to convergence. The PSO algorithm thus has tremendous potential of efficiently computing with very large numbers of concurrently operating processors.

The best continuous design point found by the PSO had an objective function of 27440 cm^3, while the best discrete solution had an objective function value of 39100 cm^3. The best results found by the PSO algorithm are compared to the theoretical answer for the continuous problem in Table 10.

Table 10 shows that the PSO algorithm found a very accurate optimum solution for the continuous design problem as compared to the theoretical answer. The best integer/discrete solution found is within the calculated upper and lower bounds summarized in Table 2. In fact the best integer/discrete solution found is equal to the upper bound shown in Table 2.

Table 10 Comparison between PSO and theoretical results

Parameter	Theoretical	PSO	
	Continuous	Continuous	Discrete
Volume	27429	27438	39100
b_1	0.5	0.5	1.0
b_2	0.5	0.5	1.0
b_3	0.5	0.5	1.0
b_4	0.5	0.5	1.0
b_5	0.5	0.5	1.0
h_1	146.27	146.39	104.0
h_2	130.85	130.93	93.0
h_3	113.31	113.39	81.0
h_4	92.54	92.58	66.0
h_5	65.44	65.47	47.0

Concluding Remarks

The PSO algorithm was applied to both a continuous and an integer/discrete structural design problem. It is shown that the PSO algorithm, which is inherently a continuous algorithm, is capable of accurately solving continuous design problems, albeit at a much higher computational cost than gradient-based optimizers.

The results show that as expected, the PSO algorithm is better suited for integer/discrete and discontinuous problems where use of a gradient-based optimizer may not be appropriate. In the present paper, it is shown that the PSO algorithm is able to solve an integer/discrete design problem more accurately and using less function evaluations as compared to a similar continuous design problem.

In terms of algorithm enhancements, the newly introduced idea of resetting the velocity vectors of violated design points has a significant positive influence on the performance of the algorithm. The craziness operator does not have a big influence. However, it seems that there might be a small advantage to combining the two enhancements as compared to just using the idea of resetting the velocity vectors.

References

[1]Michalewicz, Z. and Dasgupta, D., editors, *Evolutionary Algorithms in Engineering Applications*, Springer Verlag, 1997.

[2]Nemhauser, G. L. and Wolsey, L. A., *Integer and Combinatorial Optimization, Chapter 3*, John Wiley & Sons, 1988.

Table 8 Cost and objective function (material volume) statistics for the continuous design problem

Option	Cost				Objective			
	Mean	StdDev	Best	Worst	Mean	StdDev	Best	Worst
—	48204	50242	15900	150000	42603	24639	27440	141260
`R`	14994	4680	8700	36900	28285	2008	27442	35880
`C`	44466	49665	17100	150000	37263	17684	27442	127798
`CR`	14772	4065	8700	27900	30979	10537	27440	68071

Table 9 Cost and objective function (material volume) statistics for the integer design problem

Option	Cost				Objective			
	Mean	StdDev	Best	Worst	Mean	StdDev	Best	Worst
—	77928	63671	16500	150000	61299	21301	39100	133846
`R`	10710	2633	6600	19200	44927	12543	39100	87780
`C`	65574	60139	15600	150000	57872	22711	39100	146601
`CR`	9786	2152	6900	18000	40759	5950	39100	73108

[3]Kennedy, J. and Eberhart, R. C., "Particle Swarm Optimization," *Proceedings of the 1995 IEEE International Conference on Nerual Networks, Perth, Australia*, 1995, pp. 1942–1948.

[4]Eberhart, R. C. and Kennedy, J., "A New Optimizer Using Particles Swarm Theory," *Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan*, 1995, pp. 39–43.

[5]Kennedy, J. and Spears, W. M., "Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator," *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, Anchorage, Alaska*, May 4-9 1998.

[6]Shi, Y. and Eberhart, R. C., "A Modified Particle Swarm Optimizer," *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, Anchorage, Alaska*, May 4-9 1998.

[7]Shi, Y. H. and Eberhart, R. C., "Parameter Selection in Particle Swarm Optimization," *Evolutionary Programming VII, Lecture Notes in Computer Science*, 1998, pp. 591–600.

[8]Clerc, M., "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization," *Proceedings of the 1999 IEEE Congress on Evolutionary Computation, Washington D.C.*, 1999, pp. 1951–1957.

[9]Fourie, P. C. and Groenwold, A. A., "Particle Swarms in Size and Shape Optimization," *Proceedings of the International Workshop on Multidisciplinary Design Optimization, Pretoria, South Africa*, August 7-10 2000, pp. 97–106.

[10]Fourie, P. C. and Groenwold, A. A., "Particle Swarms in Topology Optimization," *Extended Abstracts of the Fourth World Congress of Structural and Multidisciplinary Optimization, Dalian, China*, June 4-8 2001, pp. 52–53.

[11]Vanderplaats, G. N., *Numerical Optimization Techniques for Engineering Design*, Vanderplaats Research and Development, Inc., 3rd ed., 1999.

[12]*GENESIS Version 7.0 Users Manual*, Vanderplaats Research and Development, Inc., 2001.