A Comparison of Constraint-Handling Methods for the Application of Particle Swarm Optimization to Constrained Nonlinear Optimization Problems

Genevieve Coath and **Saman K. Halgamuge**
Mechatronics and Manufacturing Research Group
Department of Mechanical and Manufacturing Engineering
University of Melbourne
[gcoath, sam]@mame.mu.oz.au

Abstract- This paper presents a comparison of two constraint-handling methods used in the application of particle swarm optimization (PSO) to constrained nonlinear optimization problems (CNOPs). A brief review of constraint-handling techniques for evolutionary algorithms (EAs) is given, followed by a direct comparison of two existing methods of enforcing constraints using PSO. The two methods considered are the application of non-stationary multi-stage penalty functions and the preservation of feasible solutions. Five benchmark functions are used for the comparison, and the results are examined to assess the performance of each method in terms of accuracy and rate of convergence. Conclusions are drawn and suggestions for the applicability of each method to real-world CNOPs are given.

1 Introduction

The optimization of constrained functions is an area of particular importance in applications of engineering, mathematics and economics, among others. Constraints can be hard or soft in nature, representing physical limits of systems or preferred operating ranges. Hence, the rigidity with which these constraints are enforced should reflect the nature of the constraints themselves.

Stochastic methods of solving constrained nonlinear optimization problems (CNOPs) are becoming popular due to the success of evolutionary algorithms (EAs) in solving unconstrained optimization problems. EAs comprise a family of computational techniques which, in general, simulate the evolutionary process [1]. Within this family, techniques may utilise population-based methods which rely on stochastic variation and selection. When applying EA techniques to CNOPs, constraints can be effectively removed via penalty functions, and the problem can then be tackled as an unconstrained optimization task.

Nonlinear optimization represents difficult challenges, as noted in [2], due to its incompatibility with general deterministic solution methods. Among many EAs investigated for solving CNOPs, particle swarm optimization (PSO) has been given considerable attention in the literature due to its speed and efficiency as an optimization technique. For real-time CNOPs, which are common in many facets of engineering for example, genetic algorithms (GAs) may be inefficient and slow, and thus PSO may provide a viable alternative. Recently, PSO has been applied to CNOPs with very encouraging results. Parsopoulos *et al* [3] compared the ability of PSO to solve CNOPs with the use of non-stationary multi-stage penalty functions to that of other EAs such as GAs. They concluded that in most cases PSO was able to find better solutions than the EAs considered. Hu and Eberhart [2], [4] implemented a PSO strategy of tracking only feasible solutions, with excellent results when compared to GAs. This paper will attempt to provide an assessment of these 2 constraint-handling methods, as to date there has been no direct comparison of their application using PSO. It should be noted that the literature is generally in agreement that in order for EAs, including PSO, to be successfully applied to real-world problems in this area, they must first prove their ability to solve highly complex CNOPs.

1.1 Constrained Nonlinear Optimization Problems

In general, a constrained nonlinear problem can be defined as follows [5]:
Minimize or maximize $f(\bar{X})$,
subject to the linear and nonlinear constraints:
$g_i(x) \geq 0, i = 1, ..., m_1$
$g_i(x) = 0, i = m_1 + 1, ..., m,$ where $x \in \Re^n$
and the bounds: $x_l \leq x \leq x_u$
Where f and $g_1, ..., g_m$ are continuously differentiable functions.

Nonlinear optimization is complex and unpredictable, and therefore deterministic approaches may be impossible or inappropriate due to the assumptions made about the continuity and differentiability of the objective function [6], [3]. Recently, there has been considerable interest in employing stochastic methods to overcome CNOPs, and this has highlighted the need for solid EA strategies for coping with infeasible solutions.

2 Particle Swarm Optimization

Particle swarm optimization was introduced by Kennedy and Eberhart in 1995 [7], [8]. It is an evolutionary computation technique which is stochastic in nature, and models itself on flocking behaviour, such as that of birds. A random initialization process provides a population of possible solutions, and through successive generations the optimum result is found. Through these generations, each "particle", or potential solution, evolves. In the global version of PSO, each particle knows its best position so far (pBest), and the best position so far among the entire group (gBest). The basic equations utilised during this evolutionary process are given below:

$$V_{id} = w \times V_{id} + c_1 rand \times (p_{id} - x_{id}) + c_2 rand \times (p_{gd} - x_{id}) \quad (1)$$

0-7803-7804-0 /03/$17.00 © 2003 IEEE

$$x_{id} = x_{id} + V_{id} \quad (2)$$

Where in Equation (1), V_{id} is the particle's new velocity, w is the inertia weight, and is set to $[0.5 + (rand/2.0)]$; p_{id} is the position at which the particle has achieved its best fitness so far, and p_{gd} is the position at which the best global fitness has been achieved so far. The acceleration coefficients c_1 and c_2 are both set to 1.49445; and V_{max} is set to the dynamic range of the particle on each dimension (problem-specific). From Equation (2), x_{id} is the particle's next position, based on its previous position and new velocity, V_{id}. The above numerical parameters, as suggested in [4], are used for all experiments in this paper. Population sizes of 30 are also used (for all but one test case, which uses 50), with the maximum number of generations being set to either 500 or 5000 depending on the complexity of the problem. PSO also has a local version, where each particle has a knowledge of the best position obtained within its own neighbourhood (lBest) instead of the best position obtained globally. This version of PSO is often used to overcome problems associated with the swarm getting trapped in local optima, which can occur in the global version of PSO when applied to certain types of problems. The references [7] and [8] provide a further explanation of PSO operation.

The application of PSO to various optimization problems has been reported extensively in the literature [7], [9], [10], with some excellent results. Whilst PSO has the advantages of speed and simplicity when applied to a wide range of problems, it is yet to be proven as a practical technique to solve CNOPs, according to [2]. Hence, the ability of PSO in dealing with constrained problems needs a thorough investigation.

3 Constraint-Handling Methods for Evolutionary Algorithms

Several methods in overcoming constraints associated with CNOPs with EAs have been reported. According to [11], these strategies fall into several distinct areas, some of which are listed below:

- Methods based on penalty functions;
- Methods based on the rejection of infeasible solutions;
- Methods based on repair algorithms;
- Methods based on specialized operators;
- Methods based on behavioural memory.

The focus of this paper is on the first two methods, which have already shown some potential when implemented with PSO. Penalty functions are a traditional means for solving CNOPs. When using this approach, the constraints are effectively removed, and a penalty is added to the objective function value (in a minimization problem) when a violation occurs. Hence, the optimization problem becomes one of minimizing the objective function and the penalty together. Penalty functions can be stationary or non-stationary. Stationary penalty functions add a fixed penalty when a violation occurs, as opposed to non-stationary penalty functions which add a penalty proportional to the amount with which the constraint was violated, and are also a function of the iteration number. Penalty functions face the difficulty of maintaining a balance between obtaining feasibility whilst finding optimality. In the case of GAs, it has been noted in [6] that too high a penalty will force the GA to find a feasible solution even it is not optimal. Conversely, if the penalty is small, the emphasis on feasibility is thereby reduced, and the system may never converge.

The rejection of infeasible solutions is distinct from the penalty function approach. Solutions are discarded if they do not satisfy the constraints of the problem. This approach has its drawbacks in that the decision must be made as to whether to consider that some infeasible solutions may be "better" than some feasible one [11]; or to distinguish between the best infeasible solutions and the worst feasible ones. This method is incorporated in evolutionary strategies (ESs), whereby infeasible members of a population are simply rejected. Recently, this method has been applied with PSO, as discussed later.

Repair algorithms seek to restore feasibility to the solution space. They have been employed widely in GAs, however it has been stated in [6] that the restoration process itself can represent a task similar in complexity to the optimization problem itself.

3.1 Constraint-Handling with Particle Swarm Optimization

3.1.1 Preservation of Feasible Solutions Method

The preservation of feasible solutions method (FSM) for constraint handling with PSO was adapted from [12] by Hu and Eberhart in [2]. When implementing this technique into the global version of PSO, the initialisation process involves forcing all particles into the feasible space before any evaluation of the objective function has begun. Upon evaluation of the objective function, only particles which remain in the feasible space are counted for the new pBest and gBest values (or lBest for the local version). The idea here is to accelerate the iterative process of tracking feasible solutions by forcing the search space to contain only solutions that do not violate any constraints. The obvious drawback of this method is that for CNOPs with extremely small feasible space, the initialization process may be impractically long, or almost impossible. Hence, no search can begin until the solution space is completely initialized.

3.1.2 Penalty Function Method

A non-stationary, multi-stage penalty function method (PFM) for constraint handling with PSO was implemented by Parsopoulos and Vrahatis in [3]. Penalty function methods require the fine-tuning of the penalty function parameters, to discourage premature convergence, whilst maintaining an emphasis on optimality. This penalty function method, when implemented with PSO, requires a rigorous checking process after evaluation of the objective function to ascertain the extent of any constraint violation/s, followed by the assignment and summation of any penalties applied.

The penalty function used in [3], originally from [13] was:

$$F(x) = f(x) + h(k)H(x), \quad x \in S \subset \Re^n \tag{3}$$

Where $f(x)$ is the original objective function to be optimized, $h(k)$ is a penalty value which is modified according to the algorithm's current iteration number, and $H(x)$ is a penalty factor, also from [3], and is defined as:

$$H(x) = \sum_{i=1}^{m} \theta(q_i(x)) q_i(x)^{\gamma(q_i(x))} \tag{4}$$

Where $q_i(x) = max\{0, g_i(x)\}, i = 1, ..., m$, $g_i(x)$ are the constraints, $q_i(x)$ is a relative violated function of the constraints; $\theta(q_i(x))$ is an assignment function, and $\gamma(q_i(x))$ is the power of the penalty function [3]. As per [3], for the experiments in this paper a violation tolerance was implemented, meaning that a constraint was only considered violated if $g_i(x) > 10^{-5}$, and the following values were used for the penalty function in all cases except Test Problem 5:

- If $q_i(x) < 1$, $\gamma(q_i(x)) = 1$, else $\gamma(q_i(x)) = 2$;
- If $q_i(x) < 0.001$, $\theta(q_i(x)) = 10$, else if $q_i(x) \leq 0.1$, $\theta(q_i(x)) = 20$, else if $q_i(x) \leq 1$, $\theta(q_i(x)) = 100$, otherwise $\theta(q_i(x)) = 300$;
- The penalty value $h(k)$ was set to $h(k) = \sqrt{k}$ for Test Problem 1 and $h(k) = k\sqrt{k}$ for the remaining test problems.

For Test Problem 5, a significantly larger value of $\theta(q_i(x))$ was used. This was established only by trial and error, due to the inability of the previous value of $\theta(q_i(x))$ to encourage any convergence.

4 Experimental Methodology

Five well-known CNOP test problems were used in our experiments, which comprise a selection of minimization problems common to those used in [3], [2], [5] and [6] for comparative purposes. The aim of the experiments was to directly compare the rate of convergence and accuracy of results of the two constraint-handling methods, in order to assess each method's suitability for application to real-world problems. The test problems used were as follows:

4.1 Test Problem 1:

$f(x) = (x_1 - 2)^2 + (x_2 - 1)^2$
Subject to:
$x_1 = 2x_2 - 1, x_1^2/4 + x_2^2 - 1 \leq 0$
Best known solution: $f^* = 1.3934651$

4.2 Test Problem 2:

$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$
Subject to:
$100 - (x_1 - 5)^2 - (x_2 - 5)^2 \leq 0,$
$(x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0,$
and the bounds: $13 \leq x_1 \leq 100, 0 \leq x_2 \leq 100$
Best known solution: $f^* = -6961.81381$

4.3 Test Problem 3:

$f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 +$
$+10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$
Subject to:
$-127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0,$
$-282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0,$
$-196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0,$
$4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0,$
and the bounds: $-10 \leq x_i \leq 10$, where $i = 1, ..., 7$
Best known solution: $f^* = 680.6300573$

4.4 Test Problem 4:

$f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 -$
-40792.141
Subject to:
$0 \leq 85.334407 + 0.005658x_2x_5 + 0.0006262x_1x_4 -$
$-0.0022053x_3x_5 \leq 92,$
$90 \leq 80.51249 + 0.0071317x_2x_5 + 0.002995x_1x_2 +$
$+0.002181x_3^2 \leq 110,$
$20 \leq 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 +$
$0.0019085x_3x_4 \leq 25,$
and the bounds: $78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45,$
$27 \leq x_i \leq 45$, where $i = 3, 4, 5$
Best known solution: $f^* = -30665.53867$

4.5 Test Problem 5:

$f(x) = x_1 + x_2 + x_3$
Subject to:
$1 - 0.0025x_4 - 0.0025x_6 \geq 0$
$1 - 0.0025x_5 - 0.0025x_7 + 0.0025x_4 \geq 0$
$1 - 0.01x_8 + 0.01x_5 \geq 0$
$1 - \frac{833.33252x_4}{x_1x_6} - \frac{100}{x_6} + \frac{83333.333}{x_1x_6} \geq 0$
$1 - \frac{1250x_5}{x_2x_7} - \frac{x_4}{x_7} + \frac{1250x_4}{x_2x_7} \geq 0$
$1 - \frac{1250000}{x_3x_8} - \frac{x_5}{x_8} + \frac{2500x_5}{x_3x_8} \geq 0$
and the bounds: $100 \leq x_1 \leq 10000, 1000 \leq x_i \leq 10000,$
where $i = 2, 3$, and $10 \leq x_j \leq 1000$, where $j = 4, 5, 6, 7, 8$
Best known solution: $f^* = 7049.24$

Twenty runs of 500 iterations were performed on each problem, and the average, best and worst solutions noted. For all problems except Test Problem 5, a population of 30 particles was used. For Test Problem 5, a population of 50 particles was used in conjunction with 5000 iterations. For each test problem, the same initialization range was used for both constraint-handling methods.

5 Results

The two constraint-handling methods, FSM and PFM, were applied to the above five test problems. Tables 1 and 2 show the best known solutions to the test problems followed by the best solution obtained by FSM and PFM respectively over the 20 runs. Tables 3 and 4 show the average optimal solution obtained by each method, followed in brackets by

Table 1: Comparison of best known solutions and best optimal solutions obtained by PSO using FSM over 20 runs.

Test	Best Known	FSM
1	1.3934651	1.3934649
2	-6961.81381	-6961.6198
3	680.630057	680.6656
4	-30665.53867	-30665.514
5**	7049.24	-

**No feasible space was initialized by FSM for this test problem.

Table 2: Comparison of best known solutions and best optimal solutions obtained by PSO using PFM over 20 runs.

Test	Best Known	PFM	(SVC)
1	1.3934651	1.393318	(0.00009)
2	-6961.81381	-6961.8214	(0.000006)
3	680.630057	680.6450	(0.000004)
4	-30665.53867	-30665.541	(0.000132)
5*	7049.24	7065.4816	(0.002407)

*A population of 50 was used and maximum iterations was set to 5000.

the average maximum attempts (MA) required to initialize a particle into the feasible space for the FSM, and the average sum of violated constraints (SVC) for the PFM. From the results, it can be seen that the accuracy of the techniques at finding near-optimal solutions is extremely competitive.
It can be seen from Table 3 that the FSM may be inappropriate for CNOPs which have an extremely small feasible space (such as Test Problem 2, whose relative size of feasible space is $< 0.0066\%$, [2]). In such cases, it can be almost impossible to randomly generate an initial set of feasible solutions. This problem was particularly apparent in Test Problem 5, where it took an inordinate amount of time to initialize less than 10% of the particles into the feasible solution space, hence the FSM was considered impractical for application to this problem. Graphical results of the convergence of each method on each test problem are shown below. It should be noted that for the sake of clarity, the graphs represent convergence over the number of iterations for which greatest convergence occurs for each test problem. Test Problem 1 was the only case where the initial values of each respective method allowed the results to be

Table 3: Average optimal solutions obtained by PSO using FSM over 20 runs.

Test	FSM	(MA)
1	1.3934650	(4)
2	-6960.4155	(60,267)
3	680.75298	(732)
4	-30665.091	(16)
5	-	-

Table 4: Average optimal solutions obtained by PSO using PFM over 20 runs.

Test	PFM	(SVC)
1	1.3934650	(0.00000)
2	-6961.4960	(0.00062)
3	680.80561	(0.10639)
4	-30648.008	(0.09383)
5	8763.7184	(0.274458)

clearly displayed on the same graph.

5.1 Test Problem 1

The rate of convergence and accuracy of both methods applied to this problem were very competitive. As can be seen in Figure 1, a rapid convergence occurs during the first 2–3 iterations. The FSM achieved an optimum result very close to the best known solution, however the average optimum results of the two methods over 20 runs were identical.

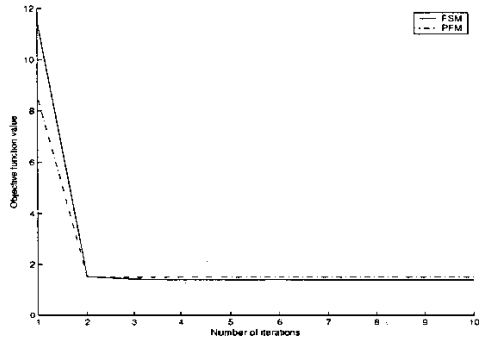

Figure 1: Convergence of FSM and PFM for Test Problem 1 over first 10 iterations

5.2 Test Problem 2

This problem is a cubic function whose relative size of feasible space is 0.0066% [2]. The PFM achieved a better optimum result and a better average optimum than the FSM, with a significantly faster convergence rate, as can be seen by comparing Figures 2 and 3.

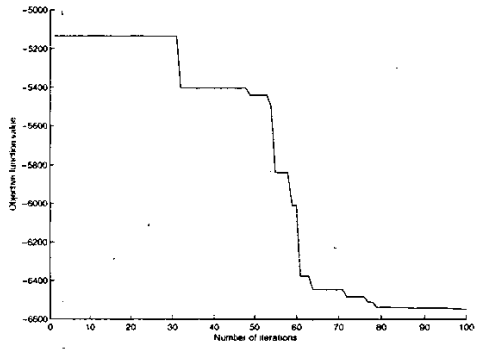

Figure 2: Convergence of FSM for Test Problem 2 over first 100 iterations

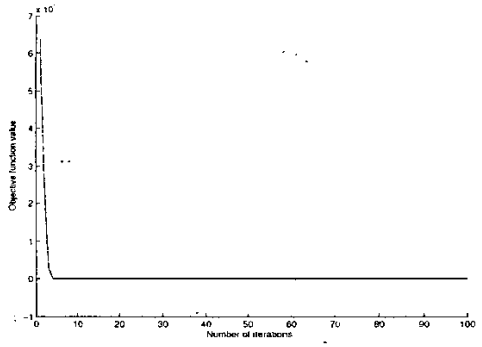

Figure 3: Convergence of PFM for Test Problem 2 over first 100 iterations

5.3 Test Problem 3

Both methods showed rapid convergence within the first 10 iterations, however the PFM showed a significantly faster rate of convergence with a slightly better optimum result. It didn't perform quite as well as the FSM with its average optimum over 20 runs. The PFM did however, perform better with this test problem than in [3], with its SVC also being lower. This different result may be attributable to the difference of PSO parameters used.

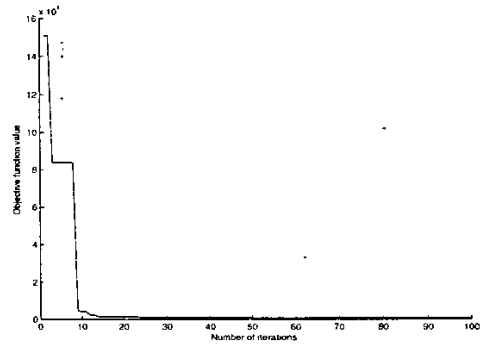

Figure 4: Convergence of FSM for Test Problem 3 over first 100 iterations

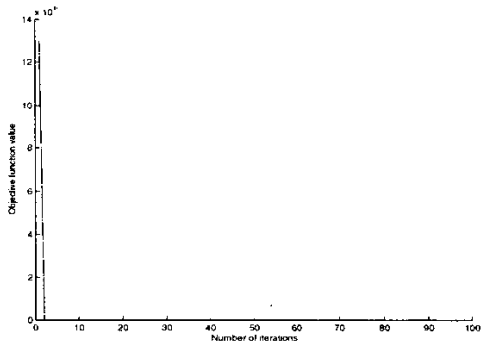

Figure 5: Convergence of PFM for Test Problem 3 over first 100 iterations

5.4 Test Problem 4

This test problem is a quadratic function which has a comparatively large relative size of feasible space (52.1230%, [2]), for which both methods showed convergence over a similar number of iterations. The PFM outperformed the FSM in terms of the optimum value found, however the FSM performed consistently well, achieving the better average optimum.

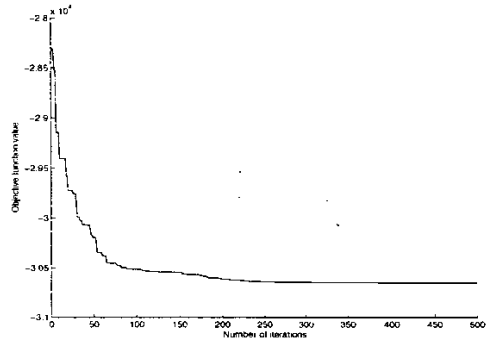

Figure 6: Convergence of FSM for Test Problem 4 over 500 iterations

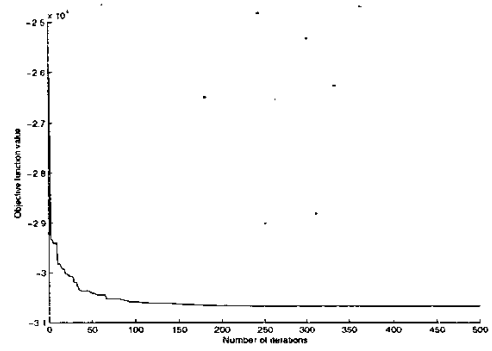

Figure 7: Convergence of PFM for Test Problem 4 over 500 iterations

5.5 Test Problem 5

For this problem, the FSM failed to initialize the search space with feasible solutions due to the binding constraints. This test problem was by far the hardest to implement for both techniques, as can be seen in Tables 2 and 4, where its optimal solution is good but its average optimal solution is comparatively poor. When implementing the PFM with this test problem, the penalty values which had worked well for all of the other test problems were changed to have a larger penalty for large violations (chosen by trial and error), to ensure convergence. Convergence did not occur with every run, hence the poor average optimum result. Proper fine-tuning of the penalty function parameters may result in a better average optimum, or implementation of the local version of PSO to overcome problems of the swarm getting stuck in local optima. Difficulties finding the optimal result for this particular test problem were also noted in [6], where GAs were used.

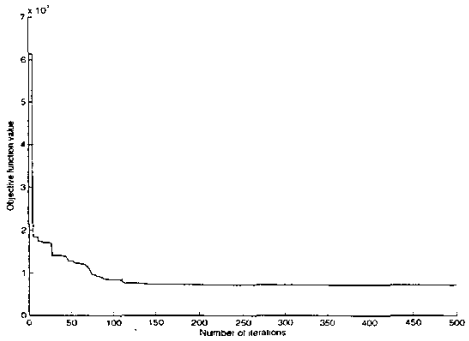

Figure 8: Convergence of PFM for Test Problem 5 over 500 iterations

6 Applications

PSO has already been applied to many real-world CNOPs; with optimization tasks as varied as power system operation [10], internal combustion engine design [14], biomedical applications [15], design of a pressure vessel and a welded beam [4]. The authors have recently applied PSO using the PFM in this paper to the optimal power flow problem, which has shown excellent preliminary results. This is a real-world CNOP which has both equality and inequality constraints. In this case, the FSM would be inappropriate and impractical due to the requirement of having to preprocess functional constraints, which would require repeated evaluation of the objective function itself during the initialization process.

7 Conclusions and Further Work

The purpose of this contribution was to assess the performance of two existing constraint-handling methods when used with particle swarm optimization (PSO) to solve constrained nonlinear optimization problems (CNOPs). This was done with a view to providing readers with some idea of the accuracy and convergence rate of each method when selecting which one to apply to real-world CNOPs. These methods were directly compared using identical swarm parameters through simulations, and the best, worst and average solutions noted. The two methods were found to be extremely competitive, however the rate of convergence of the penalty function method (PFM) was found to be significantly faster than that of the feasible solutions method (FSM) in 2 out of the 4 test problems considered. In the other 2 test problems, the rates of convergence were comparable. The fifth test problem could not be compared due to the FSM struggling to satisfy all of the constraints during the initialization process, hence the FSM was found to be impractical for this case. The accuracy of the average optimal result found by each method was identical in 1 out of the 4 test cases, and of the remaining 3, the FSM found the better average optimum result in 2 of them. Hence, the choice of constraint-handling method is very problem-dependent; some problems may require the rapid convergence characteristic shown by the PFM in some of the test cases, whereas others may have constraints which are suitable to evaluate during the initialization process. Fine-tuning of the PFM parameters may result in better average optimal solutions for the PFM, and implementation of the local version of PSO with test problems which are inclined to get stuck in local optima (such as Test Problem 5), may also improve results.

Acknowledgments

The authors would like to greatly acknowledge the Rowden White Foundation for their generous financial support of this research. The authors would also like to acknowledge the assistance of Asanga Ratnaweera for his helpful suggestions in preparing this paper.

Bibliography

[1] T. Back, D. Fogel, and T. Michalewicz, eds., *Evolutionary Computation*, vol. 1. Institute of Physics, 2000.

[2] X. Hu and R. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," in *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics*, 2002.

[3] K. Parsopoulos and M. Vrahatis, "Particle swarm optimization method for constrained optimization problems," in *Intelligent technologies — theory and applications: new trends in intelligent technologies*, vol. 76 of *Frontiers in Artificial Intelligence and Applications*, pp. 214–220, IOS Press, 2002.

[4] X. Hu, R. Eberhart, and Y. Shi. "Engineering optimization with particle swarm," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pp. 53–57, 2003.

[5] W. Hock and K. Schittkowski, *Test examples for nonlinear programming codes*, vol. 187 of *Lecture notes in economics and mathematical systems*. Springer-Verlag, 1981.

[6] J. Joines and C. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's," in *Proceedings of the First IEEE Conference on Evolutionary Computation*, vol. 2, pp. 579–584, June 1994.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimisation," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[8] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, 1995.

[9] P. Angeline, "Using selection to improve particle swarm optimization," in *Proceedings of the IEEE World Congress on Computational Intelligence*, pp. 84–89, 1998.

[10] Y. Fukuyama and H. Yoshida, "Particle swarm optimization for reactive power and voltage control in electric power systems," *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1, pp. 87–93, 2001.

[11] Z. Michalewicz, "A survey of constraint handling techniques in evolutionary computation methods," in *Proceedings of the 4th Annual Conference on Evolutionary Programming*, pp. 135–155, 1995.

[12] S. Koziel and Z. Michaelewicz, *Evolutionary algorithms homomorphous mappings and constrained parameter optimization*, vol. 7, pp. 19–44, 1999.

[13] J. Yang, Y. Chen, J. Horng, and C. Kao, *Applying family competition to evolution strategies for constrained optimization*, vol. 1213 of *Lecture notes in computer science*. Springer-Verlag, 1997.

[14] A. Ratnaweera, S. Halgamuge, and H. Watson, "Particle swarm optimization with self-adaptive acceleration coefficients," in *Proceedings of the 1st International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 264–268, 2002.

[15] R. Eberhart and H. Xiaohui, "Human tremor analysis using particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 3, 1999.