Structural and Multidisciplinary Optimization manuscript No.
(will be inserted by the editor)

Constrained Particle Swarm Optimization Using a Bi-Objective Formulation

G. Venter · **R.T. Haftka**

Received: date / Accepted: date

Abstract This paper introduces an approach for dealing with constraints when using particle swarm optimization. The constrained, single objective optimization problem is converted into an unconstrained, bi-objective optimization problem that is solved using a multi-objective implementation of the particle swarm optimization algorithm. A specialized bi-objective particle swarm optimization algorithm is presented and an engineering example problem is used to illustrate the performance of the algorithm. An additional set of 13 test problems from the literature is used to further validate the performance of the newly proposed algorithm. For the example problems considered here, the proposed algorithm produced promising results, indicating that it is an approach that deserves further consideration. The newly proposed algorithm provides performance similar to that of a tuned penalty function approach, without having to tune any penalty parameters.

Keywords Constrained particle swarm optimization · Multi-objective optimization · Composite design problem

G. Venter
Department of Mechanical and Mechatronic Engineering
Stellenbosch University
South Africa
E-mail: gventer@sun.ac.za

R.T. Haftka
Department of Mechanical and Aerospace Engineering
University of Florida
USA
E-mail: haftka@ufl.edu

This paper is based on work first presented at the Sixth International Conference on Engineering Computational Technology, Athens, Greece, 2-5 September 2008.

1 Introduction

This work introduces a specialized multi-objective particle swarm optimization (MOPSO) algorithm that is used to solve constrained, single objective optimization problems. Particle swarm optimization has received much attention in the last few years as a fairly new addition to the growing family of non-gradient global optimization algorithms. These algorithms can deal with discontinuities in the design space (e.g. numerical noise) and are easy to implement. However, these algorithms typically require many function evaluations, require parameter tuning for the specific problem at hand, and have difficulty dealing with constrained optimization problems.

The particle swarm optimization algorithm is inherently an unconstrained algorithm. To account for constraints, designers have developed many different strategies. For evolutionary algorithms a review of these strategies is provided by Coello Coello[4]. Koziel and Michalewicz[10] classifies constraint handling techniques for evolutionary algorithms as: (1) techniques that preserve feasibility, (2) techniques based on penalty functions, (3) techniques making a clear distinction between feasible and infeasible solutions and (4) other hybrid techniques. More recently Sienz and Innocente[16] classifies constraint handling strategies for particle swarm optimization as: (1) strategies that reject infeasible solutions (also known as a death penalty approach), (2) strategies that penalize infeasible solutions (also known as a penalty function approach), (3) strategies that preserve feasibility, (4) strategies that cut-off at the boundary, (5) strategies based on a bi-section approach and (6) strategies that repair infeasible solutions. Of these approaches, one of the most popular is to make use of a penalty function approach, where the objective function is penalized for any constraint violation. Penalty functions are popular because they have traditionally been used with gradient-based optimization algorithms, are general in nature and are easy to implement.

There are many different types of penalty functions available. One of the simplest and most widely used is an exterior quadratic penalty function (e.g. Vanderplaats [18]) as shown in Eq. 1

$$\overline{f}(\boldsymbol{x}) = f(\boldsymbol{x}) + r_p \sum_{i=1}^{m} max(0, g_j(\boldsymbol{x})) \tag{1}$$

where $\boldsymbol{x}$ is the vector of design variables, $f(\boldsymbol{x})$ is the original objective function, $\overline{f}(\boldsymbol{x})$ is the penalized objective function, r_p is the penalty parameter and $g_j(\boldsymbol{x})$ are the inequality constraints defined as $g_j(\boldsymbol{x}) \leq 0$. The penalty function presented in Eq. 1 has a single penalty parameter r_p that is either held constant (a static approach) or changed during the optimization (a dynamic approach). In either case the approach is problematic, since the penalty parameter has a significant impact on the performance of the algorithm, but the best choice is problem specific and can only be determined by trial and error. In addition, the approach can easily result in the algorithm converging on a local optimum design because the penalty function prevents the algorithm from traversing the infeasible design space from one feasible design to another. An example is when the constraints divide the design space into multiple island feasible regions.

More recently, adaptive penalty schemes have been introduced with the goal of eliminating any user defined, and typically problem dependent, penalty parameters. For example, Poon and Martins[13] introduced an adaptive scheme for gradient-based optimization based on the Kreisselmeier-Steinhauser function, but also taking into account the constraint sensitivities. Hamida and Schoenauer[9] introduced an adaptive scheme for evolutionary algorithms based on a population based adaptive penalty and specialized selection schemes, while Barbosa and Lemonge[2] introduced an adaptive penalty function for particle swarm optimization that automatically defines and updates different penalty parameters for each violated constraint.

A relatively new approach to constraint handling is reflected in work done by Fletcher and Leyffer[7], in which a constrained optimization problem can be considered as a bi-objective optimization problem. In this bi-objective formulation, one objective is the objective function of the original optimization problem, while the second objective is a measure of the constraint violation. Other researchers have considered the use of a multi-objective approach for handling constraints in evolutionary algorithms, but the approach is relatively new for particle swarm optimization. For example, Surry and Radcliffe[17] implemented a bi-objective genetic algorithm where a portion of the parents are selected based on the original objective function, while the remainder are selected based on a measure of the constraint violation. Although the approach still requires the user to define problem specific parameters, Surry and Radcliffe[17] mention that their method maintains the universal applicability of a penalty function, while having fewer problem dependent parameters. Zhou et al.[23] considered a bi-objective approach, using the original objective function and a measure of the constraint violation. Their approach is applied to a genetic algorithm where the Pareto strength and a minimal generation gap measure is used for selection. Venkatraman and Yen[19] introduced a two phase genetic algorithm. The first phase finds a feasible solution, by only considering the measure of constraint violation as an objective function. Once a feasible solution has been found, a bi-objective problem is defined where the original objective function and the measure of constraint violation are considered. Liu[12] considered a bi-objective approach for particle swarm optimization, but does not make use of a Pareto based multi-objective particle swarm approach to solve the resulting multi-objective optimization problem. Instead a new fitness function is defined that takes both the original objective function and a normalized measure of the constraint violation into account.

The present work presents an approach that does not have any problem dependent parameters, that is as general as a penalty function approach and that makes use of a Pareto based multi-objective particle swarm approach to solve the resulting bi-objective optimization problem. The new algorithm presented here works particularly well for optimization problems with inequality constraints. Future work will concentrate on extending the method to include efficient handling of equality constraints as well. The constrained optimization problem is first converted to a bi-objective problem, based on the work of Fletcher and Leyffer[7]. A multi-objective particle swarm optimization algorithm is then used to solve the resulting multi-objective optimization problem.

Multi-objective particle swarm optimization is a fairly new but active research field, with Reyes-Sierra and Coello Coello[15] presenting a good overview of the current state of the art within this research area. The present work starts with an existing multi-objective particle swarm optimization algorithm by Reyes-Sierra and Coello Coello[14] that appears to show good potential for solving general multi-objective problems. This algorithm is then specialized to solve constrained, single objective optimization problems using a bi-objective formulation. An engineering example is used to compare the effectiveness of both the original and the modified multi-objective algorithms with that of a penalty function based particle swarm optimization algorithm. Both an exterior quadratic penalty function, as shown in Eq. 1, as well as the adaptive penalty function introduced by Barbosa and Lemonge[2] are considered. The algorithm performance is further validated using a set of 13 test problems from the literature.

The rest of this write-up provides a quick overview of Fletcher and Leyffer's original idea, followed by a discussion on multi-objective particle swarm optimization, which also introduces the original algorithm used here. Modifications to the original algorithm are presented, followed by the example problem and the set of test problems. Finally, some concluding remarks are provided.

2 Constrained Optimization in Bi-objective Form

Fletcher and Leyffer's[7] work concentrated on sequential quadratic programming, specifically the elimination of the penalty function typically used during the one-dimensional search. They considered general, non-linear, constrained optimization problems which can be stated as

$$\begin{aligned} \text{Minimize:} \quad & f(\boldsymbol{x}) \\ \text{Such That:} \quad & \boldsymbol{g}(\boldsymbol{x}) \leq 0 \\ & \boldsymbol{x}_l \leq \boldsymbol{x} \leq \boldsymbol{x}_u \end{aligned} \tag{2}$$

where f is the objective function, $\boldsymbol{x}$ is the vector of design variables, $\boldsymbol{g}$ is the vector of inequality constraint functions and $\boldsymbol{x}_l$ and $\boldsymbol{x}_u$ are the lower and upper bounds (or side constraints) for the design variables.

In the present work the same idea introduced by Fletcher and Leyffer[7] for sequential quadratic programming, will be used to deal with constrained optimization problems within a particle swarm optimization environment. Similar to particle swarm optimization, the use of a penalty function in sequential quadratic programming is problematic. It is difficult to provide a general implementation that works for a wide range of problems, since the penalty parameters are problem dependent. Fletcher and Leyffer[7] proposed the use of a bi-objective formulation to eliminate the need of a penalty function. Their approach is based on the observation that there are two competing aims in non-linear programming. The first is to minimize the objective function f and the second is to minimize the constraint violation. These two conditions can be written as:

$$\begin{aligned} \text{Minimize:} \quad & f(\boldsymbol{x}) \\ \text{Minimize:} \quad & h(\boldsymbol{g}(\boldsymbol{x})) \end{aligned} \tag{3}$$

where $h(\boldsymbol{g}(\boldsymbol{x}))$ provides a measure of the constraint violation and is expressed as follows:

$$h(\boldsymbol{g}(\boldsymbol{x})) = \sum_{j=1}^{m} max(0, g_j(\boldsymbol{x})) \tag{4}$$

A penalty function would combine the two conditions of Eq. 3 into a single objective, unconstrained optimization problem. The bi-objective approach instead directly solves the problem as a multi-objective optimization problem. The present work will build on the idea of converting a constrained, single objective optimization problem into an unconstrained, bi-objective optimization problem within the context of particle swarm optimization. Note, however, that while a general multi-objective optimization will produce a Pareto front as the final product, for the application presented here the Pareto front is used as an intermediary, and the final result will use the point on the front with the best true objective and zero constraint violation.

3 Multi-objective Particle Swarm Optimization

Several modifications to the particle swarm optimization algorithm are needed to solve multi-objective problems. The single objective algorithm updates the position $\boldsymbol{x}$ of a particle i from the k^{th} iteration to the $(k+1)^{th}$ iteration, as follows:

$$\boldsymbol{x}^i_{k+1} = \boldsymbol{x}^i_k + \boldsymbol{v}^i_{k+1}\Delta t \tag{5}$$

The velocity vector $\boldsymbol{v}$ is updated using

$$\boldsymbol{v}^i_{k+1} = w\boldsymbol{v}^i_k + c_1 r_1 \frac{(\boldsymbol{p}^i - \boldsymbol{x}^i_k)}{\Delta t} + c_2 r_2 \frac{(\boldsymbol{p}^g - \boldsymbol{x}^i_k)}{\Delta t} \tag{6}$$

where Δt is typically taken as unity, w is known as the inertia parameter, r_1 and r_2 are random numbers between 0 and 1 and c_1 and c_2 are known as trust parameters. When solving a constrained optimization problem, a penalty function is typically used to identify the best point $\boldsymbol{p}^i$ obtained so far for each particle, as well as the best point $\boldsymbol{p}^g$ obtained so far for the swarm as a whole.

Note that the choice of $\boldsymbol{p}^g$ is referred to as a global topology, where each particle obtain information from all other particles in the group. An alternative is a local topology (e.g. Bratton and Kennedy[3]), where each particle obtains information from only a small number of other particles. For example, the Standard PSO 2007[1] algorithm randomly selects a small number of "informants" for each particle from which the best point is obtained. The best point is identified as the best point obtained so far by any of the "informants". In the present work, both the global topology outlined in Eq. 6, as well as the local topology of the Standard PSO 2007[1] algorithm were implemented. For the engineering problem, the global topology outperformed the local topology and only results for the global topology are thus presented (for comparison purposes, results from the local topology are presented in the Appendix). For the set of test problems, the best performing topology was problem dependent, and results for both topologies are presented.

The single objective algorithm uses a single best point $\boldsymbol{p}^g$ for the swarm. For multi-objective optimization, no single best point exists. Instead a number of equally good non-dominated solutions is available. (Within our bi-objective context, design point k with f_1 and f_2 is dominated by design point j if both $f_1^j \leq f_1^k$ and $f_2^j \leq f_2^k$, but not if both $f_1^j = f_1^k$ and $f_2^j = f_2^k$). Most of the multi-objective particle swarm optimization algorithms currently in circulation are Pareto-based[14], where a "best point" is identified from the available non-dominated solutions. This "best point" is referred to as a leader and each particle identifies its own leader, denoted by $\boldsymbol{p}^{gi}$. The single objective algorithm thus makes use of a single leader, while a multi-objective particle swarm algorithm (1) must identify and maintain a list of possible leaders; and (2) requires logic for selecting a leader for each particle when updating the velocity vector. Also, the logic for maintaining the best point $\boldsymbol{p}^i$ found so far by each particle must be modified. Finally, an external archive of solutions is often maintained and used to present the final result, which is a Pareto front of non-dominated solutions.

The algorithm presented here is based on the algorithm by Reyes-Sierra and Coello Coello[14]. This algorithm makes use of the crowding distance to maintain a list of leaders from which $\boldsymbol{p}^{gi}$ is selected. The crowding distance concept was introduced by Deb et al.[5] as part of the NSGA-II multi-objective genetic algorithm, which was also published as Deb et al.[6]. The crowding distance provides a density measure of non-dominated solutions surrounding a particular solution of interest. The crowding distance is obtained by first sorting the leaders according to each of the objective function values. The boundary solutions (solutions with the smallest and largest function values) are assigned crowding-distance values of infinity. All other leaders are assigned a crowding-distance value equal to the absolute normalized difference in the function values of the two nearest solutions. The process is shown graphically in Fig. 1 and outlined in Algorithm 1.

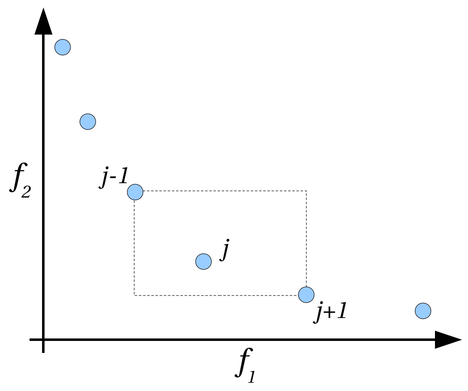

Fig. 1 Crowding distance

Algorithm 1 Crowding distance

```
1:  $l$ is the number of leaders
2:  $m$ is the number of objective functions
3:  $\xi$ is the set of leaders in matrix form
4:  $\overline{\xi}$ is the sorted set of leaders in matrix form
5:  $\overline{\xi}_{distance}$ is the crowding distance values in vector form
6:  $f_j^{min}$ is the minimum function value for the $j^{th}$ objective function
7:  $f_j^{max}$ is the maximum function value for the $j^{th}$ objective function
8:
9:  Start with the $l$ by $m$ matrix of leaders, $\xi$
10: Set all entries in $\overline{\xi}_{distance}$ to 0
11:
12: for $i = 1$ to $m$ do
13:     Sort $\xi$ according to column $i$ to obtain $\overline{\xi}$
14:     Set crowding distance $\overline{\xi}[1]_{distance} = \overline{\xi}[l]_{distance} = \infty$
15:     for $j = 2$ to $(l-1)$ do
16:         $\overline{\xi}[j]_{distance} = \overline{\xi}[j]_{distance} + \frac{\overline{\xi}[j+1][i] - \overline{\xi}[j-1][i]}{f_j^{max} - f_j^{min}}$
17:     end for
18: end for
```

The key features of the multi-objective particle swarm algorithm presented by Reyes-Sierra and Coello Coello[14] are summarized in Algorithm 2. The algorithm maintains a list of leaders that consists of a subset of non-dominated designs found so far. The number of leaders can quickly grow very large and as a result most multi-objective particle swarm optimization algorithms limit the number of leaders that is stored. Reyes-Sierra and Coello Coello[14] limits the number of leaders to be no more than the swarm size, by saving only the non-dominated solutions with the best (largest) crowding distance values. For each particle, a leader is selected to act as $\boldsymbol{p}^{gi}$ based on a binary tournament. The tournament selects two random leaders from the list of available leaders. The leader with the best (largest) crowding distance is the winner of the tournament and is selected as $\boldsymbol{p}^{gi}$. In addition, the best point $\boldsymbol{p}^i$ found so far for each particle is updated only if a new point dominates the current best point for that particle, or if both points are non-dominated with respect to each other.

Reyes-Sierra and Coello Coello[14] implements mutation by dividing the swarm into three equal parts, with a different mutation operator applied to each part. In the present work, a single mutation operator is applied to the whole swarm. For each particle in each iteration, the mutation operator has a 10% probability of changing the position of the particle to a random position in the design space. After the optimization is completed, a filter is applied to the external archive to extract the Pareto front. However, as discussed in the next section, the Pareto front is not required for solving single objective constrained optimization problems.

The original algorithm outlined in Algorithm 2 was implemented and tested on an unconstrained, bi-objective test case from Deb et al.[6]. The test case can be summarized as:

Algorithm 2 Multi-objective particle swarm optimization algorithm

```
 1: Initialize swarm
 2: Identify leaders (non-dominated solutions)
 3: Save leaders to external archive
 4: Calculate crowding distance for all leaders
 5: while Iter less than MaxIter do
 6:     for Each Particle do
 7:         Select leader (binary tournament)
 8:         Update position and velocity
 9:         Apply mutation
10:         Perform function evaluation
11:         Update best point $\boldsymbol{p}^i$
12:     end for
13:     Update leaders
14:     Save leaders to external archive
15:     Calculate crowding distance for all leaders
16: end while
17: Post-process external archive
```

$$
\begin{aligned}
f_1(x) &= x^2 \\
f_2(x) &= (x-2)^2 \qquad (7) \\
x &\in [-100, 100]
\end{aligned}
$$

The Pareto front for this problem is well known. It is a convex curve with $x \in [0, 2]$. The results found from the algorithm implemented in the present work are shown in Fig. 2 and corresponds well to the results presented in Deb et al.[6]. The results presented in Fig. 2 where obtained with a swarm size of 20 particles and 40 iterations.

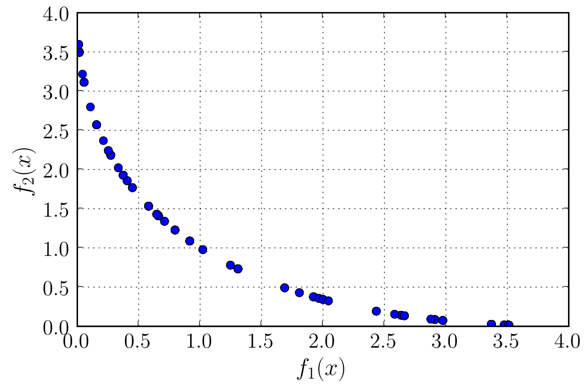

Fig. 2 Bi-objective example problem

4 Specialization of the Basic Algorithm

The algorithm outlined in Section 3 can be used as is to solve single objective constrained optimization problem using Fletcher and Leyffer's approach as summarized in Eq. 3. However, the algorithm can be improved by specializing it to the problem at hand. First, the formulation will always result in a bi-objective problem, regardless of the number of constraints. Second, the full Pareto front is not of interest. The only region of interest is the area where the constraint violation $h(\boldsymbol{g}(\boldsymbol{x}))$ is small. The optimum solution will be the non-dominated solution with the smallest $h(\boldsymbol{g}(\boldsymbol{x}))$ value. This will either be the most feasible point, if no feasible solution is found, or the feasible solution with the smallest objective function value. If the original objective function is shown on the abscissa and the $h(\boldsymbol{g}(\boldsymbol{x}))$ value on the ordinate of Fig 2, the solution to the original optimization problem will be the rightmost point, where $h(\boldsymbol{g}(\boldsymbol{x}))$ is a minimum. Note that the Pareto front, especially in the region where $h(\boldsymbol{g}(\boldsymbol{x}))$ is small, could be of significant interest to the designer for performing trade-off studies to immediately judge the impact of constraint violations on the objective function value.

4.1 Leaders based on constraint violation

Many multi-objective optimization algorithms have the goal of providing an answer that fully covers the Pareto front. The algorithm outlined in Section 3 makes use of the crowding distance to achieve this goal. First the crowding distance is used to maintain the list of leaders, and secondly it is used to select a leader for each particle when calculating the velocity vector. In the present work, the Pareto front is still important, but instead of covering the full Pareto front equally well, the goal is to concentrate on the area where $h(\boldsymbol{g}(\boldsymbol{x}))$ is small.

The original algorithm can easily be modified to achieve this new goal by using the $h(\boldsymbol{g}(\boldsymbol{x}))$ value instead of the crowding distance to both maintain the list of leaders and to select a leader for each particle when calculating the velocity vector. The modifications can be summarized as follows:

1. The number of leaders are still limited to the swarm size with the number of leaders constrained based on their $h(\boldsymbol{g}(\boldsymbol{x}))$ values. Smaller $h(\boldsymbol{g}(\boldsymbol{x}))$ values are preferred.
2. The leader for each particle is selected from a binary tournament based on the $h(\boldsymbol{g}(\boldsymbol{x}))$ values. The leader with the smallest $h(\boldsymbol{g}(\boldsymbol{x}))$ value wins the tournament and is selected as the leader for the particle.

To illustrate the difference between the two algorithms, the example problem of Eq. 7 was solved with the modified algorithm as outlined in this section. The example problem can be considered as a bi-objective representation of a single objective constrained optimization problem. In this case, $f_1(x)$ represents the original objective function and $f_2(x)$ the measure of constraint violation $h(\boldsymbol{g}(\boldsymbol{x}))$. When using the modified algorithm where the crowding distance is replaced with the constraint violation, a higher density of points is expected in the area where $f_2(x)$ is small. The results are presented in Fig. 3 and clearly illustrates a higher density of points in the area where $f_2(x)$ is small. The results presented in Fig. 3 where obtained with a swarm size of 20 particles and 40 iterations.

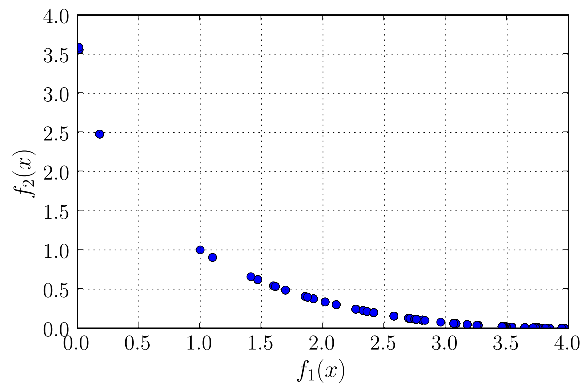

Fig. 3 Bi-objective example problem with constraint violation used to choose leaders

4.2 Two criteria for selecting leaders

The specialized algorithm outlined here was tested on several test problems with good results. In all test cases considered, the specialized algorithm clearly outperformed the original multi-objective particle swarm optimization algorithm outlined in Section 3. However, it was noticed that if only the constraint violation is used to maintain the list of leaders, that the algorithm can quickly deteriorate to having all the leaders be extremely close to the most feasible point. The result is that the algorithm quickly converges to a small number of leaders, and many times to a single leader. This loss of diversity among the leaders helps the algorithm to quickly converge to the feasible space, but has the drawback that the algorithm easily gets trapped in a local minimum in cases where the feasible region is non-convex or divided into multiple regions.

To overcome this limitation, the specialized algorithm was slightly modified to help promote diversity in the list of leaders. At the end of each design iteration the list of non-dominated solutions is considered and the best subset is stored as the list of available leaders. In the original algorithm the best subset is identified based on the crowding distance, in the specialized algorithm the selection is done based on the constraint violation.

Three variations of the specialized algorithm were considered that identify the subset of non-dominated solutions based on two selection criteria instead of just one. The goal is to identify both leaders that have a small constraint violation, and leaders that may have other attractive features, for example a large crowding distance. In all cases, the first selection criterion is the constraint violation as before. The three variations thus only differ in the second criterion, with the following criteria considered:

1. The objective function value
2. The crowding distance value (larger is better)
3. A randomly selected non-dominated design

The list of leaders is compiled from the available non-dominated solutions found in the current iteration as well as those previously included in the list of leaders. When using two selection criteria for updating the list of leaders, a random number generator is used to select which of the two criteria will be used to identify the next leader. The current implementation makes use of a 75% probability of selecting the next leader using the smallest constraint violation value and a 25% probability of selecting the next leader using one of the alternative criteria as outlined above. The net effect of all three variations is to promote diversity in the list of leaders.

5 Engineering Example

An engineering example problem is presented to illustrate the performance of the newly presented algorithms. Both the original and specialized versions of the multi-objective particle swarm optimization algorithm were tested to evaluate the effectiveness of each. The two multi-objective particle swarm optimization algorithms were also compared to a single objective particle swarm optimization algorithm that makes use of a penalty function approach.

The example problem presented, is a variation of a composite laminate design problem presented in Grosset et al.[8]. The problem is formulated in terms of the laminate parameters and the optimization problem is defined as finding n continuous ply angle and corresponding ply thickness values that will maximize the transverse in-plane stiffness coefficient A_{22} for a symmetric and balanced composite lay up of total thickness h. The design is subjected to a constraint on the effective Poisson's ratio ν_{eff} and constraints that limit the ply angles to fall within one of three ranges. The problem can be summarized as

$$\begin{aligned}
\textit{Maximize}: &\quad A_{22} = h\left(U_1 - U_2 V_1^* + U_3 V_3^*\right)\\
\textit{SuchThat}: &\quad 0.48 \le \nu_{eff} \le 0.52\\
&\quad -5^\circ \le \theta_k \le 5^\circ \quad \text{or}\\
&\quad 40^\circ \le \theta_k \le 50^\circ \quad \text{or}\\
&\quad 85^\circ \le \theta_k \le 95^\circ\\
&\quad 0.001 \le t_k \le 0.05
\end{aligned} \tag{8}$$

where

$$V^*_{\{1,3\}} = \frac{2}{h}\int_0^{\frac{h}{2}} \{cos2\theta, cos4\theta\}\,dz = \frac{2}{h}\sum_{k=1}^{n} t_k\{cos2\theta_k, cos4\theta_k\} \tag{9}$$

$$\nu_{eff} = \frac{A_{11}}{A_{22}} = \frac{U_4 - U_3V_3^*}{U_1 - U_2V_1^* + U_3V_3^*} \tag{10}$$

and θ_k represent the ply angles, t_k the ply thicknesses (in inches) and the U_i values are material invariants as summarized in Table 1. For the example problem considered here, $n = 3$ was used, resulting in 3 ply orientation θ_k and 3 ply thickness t_k design variables (a total of 6 design variables).

Table 1 Material properties for graphite epoxy

Parameter	Value
U_1	0.8897×10^7 psi
U_2	1.0254×10^7 psi
U_3	0.2742×10^7 psi
U_4	0.3103×10^7 psi

The problem was solved using a single objective particle swarm optimization algorithm, the multi-objective particle swarm optimization algorithm of Reyes-Sierra and Coello Coello[14] and the specialized bi-objective particle swarm optimization algorithm presented here. The single objective particle swarm optimization algorithm made use of an exterior quadratic penalty function as shown in Eq 1 and the adaptive penalty method of Barbosa and Lemonge[2]. The adaptive penalty method of Barbosa and Lemonge provides a penalized objective function as follows

$$F(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \text{if } \boldsymbol{x} \text{ is feasible} \\ \overline{f}(\boldsymbol{x}) + \sum_{j=1}^{m} k_j v_j(\boldsymbol{x}) & \text{otherwise} \end{cases} \tag{11}$$

where $f(\boldsymbol{x})$ is the original objective function,

$$\overline{f}(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \text{if } f(\boldsymbol{x}) > \langle f(\boldsymbol{x})\rangle \\ \langle f(\boldsymbol{x})\rangle & \text{otherwise} \end{cases}, \tag{12}$$

$$k_j = |\langle f(\boldsymbol{x})\rangle| \frac{\langle v_j(\boldsymbol{x})\rangle}{\sum_{i=1}^{m}[\langle v_i(\boldsymbol{x})\rangle]^2} \tag{13}$$

the $\langle\,\rangle$ operator indicates the mean for the population and $v_j(\boldsymbol{x}) = max(0, g(\boldsymbol{x}))$.

In all cases, 100 optimization runs were performed using swarms with 30 particles applied over 100 iterations. The probability of applying mutation was 10% and w=0.5, c_1=1.75 and c_2=2.25 values were used. These parameters were not tuned for the specific problem considered here. Instead, values were selected based on previous experience [20] [21] [22] with the single objective particle swarm algorithm implemented here (which was also the basis for the two multi-objective particle swarm algorithms). The same parameters and number of function evaluations were used for all algorithms, and their variations, in the following comparison study.

The influence of the penalty parameter on the performance of the single objective particle swarm optimization algorithm is illustrated in Fig. 4, where the optimization was repeated for penalty parameter values of 1E4, 1E6, 1E8 and 1E10 respectively. Figure 4 summarizes the results of all 100 independent optimization runs that were performed. The figure contains only results for the cases where feasible solutions were found, sorted in descending order. For example, for the 1E4 case, 39 of the 100 optimization runs were able to find a feasible solution, with roughly 12 runs finding values close the global optimum of $A_{22} = 1.25 \times 10^6$. Figure 4 shows that, as expected, the value of the penalty parameter has a significant influence on the performance of the algorithm, with the best performing value equal to 1E8.

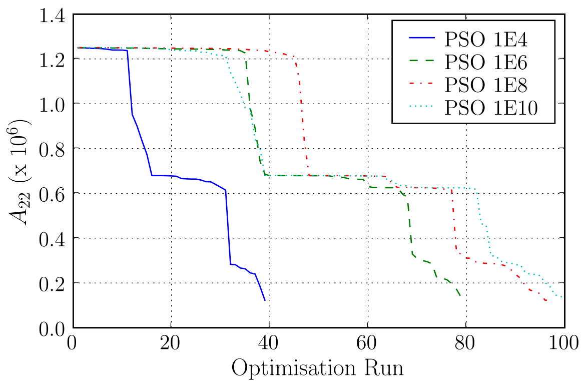

Fig. 4 Influence of the penalty parameter on the single objective particle swarm optimization algorithm as tested on the engineering example problem

Figure 5 illustrates the effect of promoting diversity in the list of leaders. Using the objective function value as a second selection criterion decreased the effectiveness of the algorithm. However, using either the crowding distance or a random non-dominated design significantly increased the effectiveness of the algorithm. The poor performance of using the objective function value as a second selection criterion is to be expected. Using the constraint violation and the objective function values as selection criteria will only select leaders from one of the two extreme points of the Pareto front. However, using the constraint violation and either the crowding distance or random non-dominated design, will include leaders distributed along the Pareto front.

Figure 6 provides a comparison of the best variants of each algorithm. Figure 6 shows the results obtained from

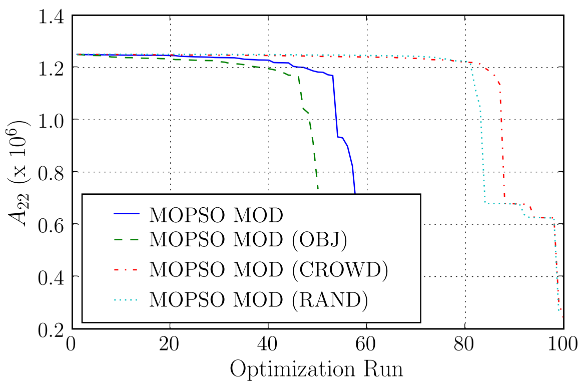

Fig. 5 Modified multi-objective particle swarm optimization algorithm variants as tested on the engineering example problem

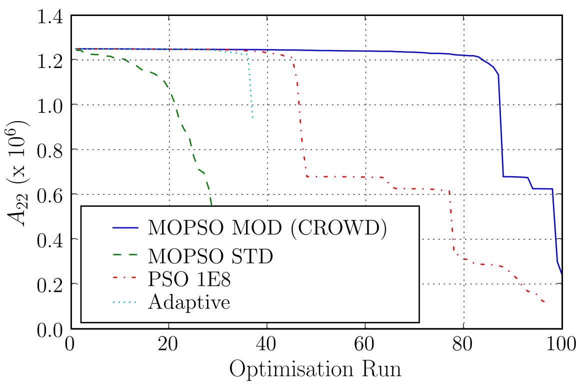

Fig. 6 Comparison of best variation of each algorithm as tested on the engineering example problem

the original multi-objective algorithm, the specialized bi-objective algorithm using the crowding distance as a second selection criterion and the single objective particle swarm optimization algorithm using both a quadratic exterior penalty function (with r_p = 1E8) as well as the adaptive penalty function of Barbosa and Lemonge[2]. From Fig. 6 it is clear that the newly proposed bi-objective algorithm provides the best performance for the problem considered. This algorithm had a 100% success rate of finding feasible designs and more than an 80% success rate of finding designs close the global optimum. Of the remaining algorithms, the exterior quadratic penalty function provided the best performance after the penalty parameter was tuned. The exterior quadratic penalty function had a 97% success rate of finding feasible designs, but less than a 45% success rate of finding designs close to the global optimum. The adaptive penalty function of Barbosa and Lemonge[2] was very successful at finding the global optimum and did not get caught in the local minimum at all. However, the algorithm only found a feasible solution in 37 of the 100 optimization runs. The original multi-objective algorithm of Reyes-Sierra and Coello Coello[14] performed the worst. The best results obtained from each of the four algorithms are summarized in Table 2.

6 Performance Validation

The performance results obtained for the engineering example problem were further validated using a standard set of test problems from the literature. The test problems selected were obtained from Liang et al.[11]. The complete set consists of 24 problems, from which all single objective problems with only inequality constraints were selected. This process resulted in a test set consisting of 13 problems. Future work will concentrate on expanding the algorithm presented here to efficiently deal with equality constraints as well.

As for the engineering example problem, the solution of each problem was repeated 100 times. Based on the results obtained for the engineering example problem, the following six algorithms were considered:

1. The modified multi-objective particle swarm optimization algorithm, using the crowding distance as the second selection criterion.
2. The original multi-objective particle swarm optimization algorithm of Reyes-Sierra and Coello Coello[14].
3. The single objective particle swarm optimization algorithm using a local topology and a fixed penalty parameter.
4. The single objective particle swarm optimization algorithm using a global topology and a fixed penalty parameter.
5. The single objective particle swarm optimization algorithm using a local toplogy and the adaptive penalty scheme of Barbosa and Lemonge[2].
6. The single objective particle swarm optimization algorithm using global toplogy and the adaptive penalty scheme of Barbosa and Lemonge[2].

For the single objective particle swarm optimization algorithms, only the results for the best penalty parameters are shown. For each problem, the best penalty parameter was selected from 1E4, 1E6, 1E8, 1E10 and 1E12. In all cases, the same algorithm parameters were used as outlined for the engineering example problem, except for the number of particles and the number of design iterations. To account for the increased number of design variables, the number of particles was increased from 30 to 50 and the number of design iterations from 200 to 500.

The modified multi-objective particle swarm optimization algorithm and the two single objective particle swarm optimization algorithms using either a local or a global topology with a fixed (but tuned) penalty parameter clearly out performed the other algorithms for the 13 test problems considered. As a result, only the results obtained from these

Table 2 Optimization results for the engineering example problem

Parameter	PSO		MOPSO	Modified MOPSO
	1E8	Adaptive		Crowding
Laminate (Degrees)	$[\pm 95, \pm 44.3, \pm 44.5]_s$	$[\pm 95, \pm 44.3, \pm 44.5]_s$	$[\pm 95, \pm 43.7, \pm 42.3]_s$	$[\pm 95, \pm 44.8, \pm 44.4]_s$
Thickness (in)	$[0.0304, 0.05, 0.05]_s$	$[0.0304, 0.05, 0.05]_s$	$[0.0323, 0.05, 0.05]_s$	$[0.030, 0.05, 0.05]_s$
ν_{eff}	0.4800	0.4800	0.4800	0.4800
Feasible	97/100	37/100	29/100	100/100
Best	1.2505×10^6	1.2506×10^6	1.2434×10^6	1.2503×10^6
Worst	0.1226×10^6	0.9287×10^6	0.5005×10^6	0.2395×10^6
Mean	0.8504×10^6	1.2373×10^6	1.0585×10^6	1.1551×10^6
StdDev	0.4010×10^6	0.0519×10^6	0.2104×10^6	0.2231×10^6

Table 3 Results for the test problems from obtained from Liang et al.[11]

Problem				Results						
ID	NDVAR	NCONSTR	Best Known	Algorithm	Penalty Parameter	Success Rate (%)	Best Obj	Worst Obj	Mean Obj	Std Dev Obj
g01	13	9	-15.00	PSO (lbest)	1.E12	100	-14.95	-5.00	-9.76	2.81
				PSO (gbest)	1.E12	100	-14.81	-3.00	-7.48	2.26
				MOPSO (mod)	-	100	-14.98	-6.00	-10.38	2.54
g02	20	2	-0.804	PSO (lbest)	1.E4	100	-0.473	-0.224	-0.330	0.041
				PSO (gbest)	1.E12	94	-0.637	-0.314	-0.475	0.078
				MOPSO (mod)	-	100	-0.700	-0.373	-0.539	0.065
g04	5	6	-30666	PSO (lbest)	1.E12	100	-30665	-30663	-30665	0.568
				PSO (gbest)	1.E12	93	-30666	-30184	-30639	108.2
				MOPSO (mod)	-	100	-30666	-30656	-30664	1.892
g06	2	2	-6962	PSO (lbest)	1.E12	100	-6960	-6784	-6942	25.30
				PSO (gbest)	1.E12	96	-6962	-6745	-6956	27.61
				MOPSO (mod)	-	100	-6959	-6910	-6939	12.68
g07	10	8	24.31	PSO (lbest)	1.E12	100	30.64	62.31	40.03	5.437
				PSO (gbest)	1.E10	100	25.72	124.8	32.52	10.98
				MOPSO (mod)	-	100	26.97	72.54	36.96	8.934
g08	2	2	-0.096	PSO (lbest)	1.E4	100	-0.096	-0.096	-0.096	0.000
				PSO (gbest)	1.E4	100	-0.096	-0.096	-0.096	0.000
				MOPSO (mod)	-	100	-0.096	-0.095	-0.096	0.000
g09	7	4	680.6	PSO (lbest)	1.E12	100	682.0	690.4	684.9	1.502
				PSO (gbest)	1.E8	100	680.8	685.8	681.9	0.920
				MOPSO (mod)	-	100	681.8	693.5	685.4	2.282
g10	8	6	7049	PSO (lbest)	1.E10	100	7645	10250	8929	471.0
				PSO (gbest)	1.E12	95	7211	18706	9751	2546
				MOPSO (mod)	-	91	7611	15553	8992	1225
g12	3	1	-1.000	PSO (lbest)	1.E4	100	-1.000	-1.000	-1.000	0.000
				PSO (gbest)	1.E4	100	-1.000	-1.000	-1.000	0.000
				MOPSO (mod)	-	100	-1.000	-1.000	-1.000	0.000
g16	5	38	-1.905	PSO (lbest)	1.E12	100	-1.899	-1.880	-1.890	0.004
				PSO (gbest)	1.E10	98	-1.905	-1.415	-1.859	0.128
				MOPSO (mod)	-	98	-1.896	-1.815	-1.872	0.012
g18	9	13	-0.866	PSO (lbest)	1.E8	100	-0.758	-0.350	-0.556	0.080
				PSO (gbest)	1.E10	99	-0.859	-0.448	-0.655	0.130
				MOPSO (mod)	-	99	-0.756	-0.057	-0.552	0.110
g19	15	5	32.66	PSO (lbest)	1.E10	100	35.82	102.1	56.77	15.59
				PSO (gbest)	1.E12	100	36.14	547.9	99.64	89.76
				MOPSO (mod)	-	100	34.35	192.3	63.30	25.17
g24	2	2	-5.508	PSO (lbest)	1.E8	100	-5.508	-5.506	-5.507	0.000
				PSO (gbest)	1.E10	100	-5.508	-5.508	-5.508	0.000
				MOPSO (mod)	-	100	-5.508	-5.503	-5.506	0.001

ID *is the problem designation from Liang et al.[11],* ***NDVAR*** *is the number of design variables,* ***NCONSTR*** *is the number of inequality constraints,* ***Best Known*** *is the best known solution as reported by Liang et al.[11]*

three algorithms are presented. The results are presented in Table 3, which provides the problem designation from Liang et al.[11], the number of design variables and inequality constraints, the best known solution from Liang et al.[11] and the results obtained here. For the algorithms considered here, the best penalty parameter, the success rate of finding feasible solutions (out of 100 optimizations) and the best, worst, mean and standard deviation of the objective function values (for the feasible solutions found) are provided.

Table 3 clearly illustrates that all three algorithms are very successfull at finding feasible solutions, with the success rate never dipping below 90%. Also, all three algorithms are competitive in terms of the mean objective function value of the feasible solutions found. Clearly the newly proposed algorithm performs well when compared to the fixed penalty parameter algorithms, but without the need of tuning the problem specific penalty parameter.

7 Conclusion

This paper presents a bi-objective formulation for solving single objective, constrained optimization problems using a specialized multi-objective particle swarm optimization algorithm. This approach is presented as an alternative for using a penalty function approach when solving constrained optimization problems by particle swarm optimization. A multi-objective particle swarm optimization algorithm from the literature is implemented and modified to specifically solve the bi-objective problem of interest. A composite laminate design problem is solved to demonstrate the effectiveness of the approach and to compare the original and modified multi-objective particle swarm optimization algorithms. Results from a single objective particle swarm optimization algorithm implementing both a quadratic exterior penalty function and an adaptive penalty function are also presented for comparison. The example illustrates that the proposed algorithm provides performance that is similar to that of a tuned penalty function approach, within the need for tuning the penalty parameter.

Variations that improve the diversity in the list of leaders of the specialized bi-objective particle swarm optimizer were also investigated. Using both the constraint violation and the crowding distance as selection criteria resulted in the best performing algorithm.

The results observed from the engineering example problem were further validated with a set of 13 test problems selected from the literature. Based on the results obtained from the example problems considered here, the proposed algorithm does seem promising enough to validate further consideration as an alternative approach for inequality constraint handling within a particle swarm environment. The results presented indicate that the modified multi-objective particle swarm optimization algorithm provide performance that is competitive to that obtained from a penalty function implementation, with the benefit that no tuning of the constraint handling logic is required. Future work will expand the proposed method to include the efficient handling of equality constraints as well.

Acknowledgements This work has been supported in part by the NASA Constellation University Institute Program (CUIP) and by the National Research Foundation (NRF) of South Africa. Any opinion, findings and conclusions or recommendations expressed in this material are those of the author(s) and therefore the NRF does not accept any liability in regard thereto.

References

1. URL http://www.particleswarm.info
2. Barbosa, H., Lemonge, A.: A New Adaptive Penalty Scheme for Genetic Algorithms. Information Sciences **156**(3–4), 215–251 (2003)
3. Bratton, D., Kennedy, J.: Defining a Standard for Particle Swarm Optimization. In: Proceedings of the 2007 IEEE Swarm Intelligence Symposium, pp. 120–127 (2007)
4. Coello Coello, C.: Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art. Computer Methods in Applied Mechanics and Engineering **191**(11–12), 1245–1287 (2002)
5. Deb, K., Agarwal, S., Pratap, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In: Proceedings of the Parallel Problem Solving from Nature VI Conference, pp. 849–858 (2000)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002)
7. Fletcher, R., Leyffer, S.: Nonlinear Programming without a Penalty Function. Mathematical Programming **91**(2), 239–269 (2002)
8. Grosset, L., LeRiche, R., Haftka, R.T.: A Double-Distributed Statistical Algorithm for Composite Laminate Optimization. Structural and Multidisciplinary Optimization **31**(1), 49–59 (2005)
9. Hamida, B., Schoenauer, M.: An Adaptive Algorithm for Constrained Optimization Problems. In: Proceedings of the 6th Conference on Parallel Problems Solving from Nature, pp. 529–539 (2000)
10. Koziel, S., Michalewicz, Z.: Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. Evolutionary Computation **7**(1), 19–44 (1999)
11. Liang, J., Runarsson, T., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello Coello, C., Deb, K.: Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical report, Nanyang Technological University, Singapore (2006)
12. Liu, C.A.: New multiobjective pso algorithm for nonlinear constrained programming problems. In: Advances in Cognitive Neurodynamics ICCN 2007, pp. 955–962 (2007)
13. Poon, N., R.R.A.M., J.: An Adaptive Approach to Constraint Aggregation Using Adjoint Sensitivity Analysis. Structural and Multidisciplinary Optimization **34**(1), 61–73 (2007)
14. Reyes-Sierra, M., Coello Coello, C.: Improving PSO-based Multi-Objective Optimization Using Crowding, Mutation and ε-dominance. In: Third International Conference on Evolutionary Multi-Criterion Optimization, Guanajuato, Mexico, pp. 505–519 (2005)

15. Reyes-Sierra, M., Coello Coello, C.: Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. International Journal of Computational Intelligence Research **2**(3), 287–308 (2006)
16. Sienz, J., Innocente, M.: Trends in Engineering Computational Technology, chap. Particle Swarm Optimization: Fundamental Study and its Application to Optimization and to Jetty Scheduling Problems, pp. 103–126. Saxe-Coburg Publications (2008)
17. Surry, P., Radcliffe, N.: The COMOGA Method: Constrained Optimisation by Multi-Objective Genetic Algorithms. Control and Cybernetics **26**(3) (1997)
18. Vanderplaats, G.N.: Numerical Optimization Techniques for Engineering Design, 4rd edn. Vanderplaats Research and Development, Inc., 1767 S. 8th St., Suite 100, Colorado Springs, CO (2005)
19. Venkatraman, S., Yen, G.: A Generic Framework for Constrained Optimization Using Genetic Algorithms. IEEE Transactions on Evolutionary Computation **9**, 424–435 (2005)
20. Venter, G., Sobieszczanski-Sobieski, J.: Particle Swarm Optimization. AIAA Journal **41**(8), 1583–1589 (2003)
21. Venter, G., Sobieszczanski-Sobieski, J.: Multidisciplinary Optimization of a Transport Aircraft Wing using Particle Swarm Optimization. Structural and Multidisciplinary Optimization **26**(1), 121–131 (2004)
22. Venter, G., Sobieszczanski-Sobieski, J.: A Parallel Particle Swarm Optimization Algorithm Accelerated by Asynchronous Evaluations. Journal of Aerospace Computing, Information, and Communication **3**(3), 123–137 (2006)
23. Zhou, Y., Li, Y., He, J., Kang, L.: Multi-Objective and MGG Evolutionary Algorithm for Constrained Optimization. In: The 2003 Congress on Evolutionary Computation, pp. 1–5 (2003)

Appendix
Local versus Global Topology Study

When implementing the black single objective particle swarm optimization algorithm, either a local or a global topology can be selected for updating the velocity vector. According to Bratton and Kennedy[3], the local topology is generally preferred since it helps to avoid local minima. However, Bratton and Kennedy[3] also states that despite the advantages of a local topology, it is important to note that it should not always be considered as the optimal choice for all problems. In the present work, both the ability of finding feasible designs as well as the ability of finding the global optimum are compared. As a result, the local topology was tested against the global topology to ensure that the best selection is made for the example problem at hand.

The local topology implemented was obtained from the Standard PSO 2007[1] algorithm. This local topology randomly selects a small number of “informants” for each particle from which the best point is obtained. The best point is identified as the best point obtained so far by any of the “informants”. Figure 4 provides the results obtained from the global topology outlined in Eq. 6 (results for the adaptive penalty scheme are provided in Fig. 6). Figure 7 below provides comparative results obtained from the local topology implementation.

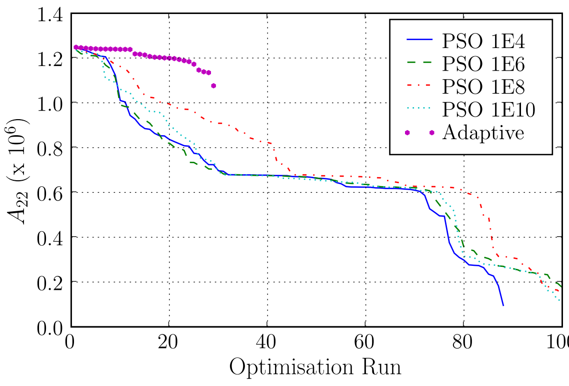

Fig. 7 Penalty function results with local topology as tested on the engineering example problem

When comparing Figs. 4, 6 and 7, it is clear that the global topology consistently outperforms the local topology for the engineering example problem considered here.