Engineering Optimization
Vol. 39, No. 1, January 2007, 49–68

Taylor & Francis
Taylor & Francis Group

An efficient multi-objective optimization algorithm based on swarm intelligence for engineering design

M. JANGA REDDY and D. NAGESH KUMAR*

Department of Civil Engineering, Indian Institute of Science, Bangalore-560 012, India

(Received 22 September 2005; revised 31 March 2006; in final form 8 June 2006)

As there is a growing interest in applications of multi-objective optimization methods to real-world problems, it is essential to develop efficient algorithms to achieve better performance in engineering design and resources optimization. An efficient algorithm for multi-objective optimization, based on swarm intelligence principles, is presented in this article. The proposed algorithm incorporates a Pareto dominance relation into particle swarm optimization (PSO). To create effective selection pressure among the non-dominated solutions, it uses a variable size external repository and crowding distance comparison operator. An efficient mutation strategy called elitist-mutation is also incorporated in the algorithm. This strategic mechanism effectively explores the feasible search space and speeds up the search for the true Pareto-optimal region. The proposed approach is tested on various benchmark problems taken from the literature and validated with standard performance measures by comparison with NSGA-II, one of the best multi-objective evolutionary algorithms available at present. It is then applied to three engineering design problems. The results obtained amply demonstrate that the proposed approach is efficient and is able to yield a wide spread of solutions with good coverage and convergence to true Pareto-optimal fronts.

Keywords: Multi-objective optimization; Swarm intelligence; Particle swarm optimization; Elitist-mutation; Pareto-optimal solutions; Engineering design

1. Introduction

Multi-objective optimization techniques play an important role in engineering design, resources optimization, and many other fields. Their main purpose is to find a set of best solutions from which a designer or a decision-maker can choose a solution to derive maximum benefit from the available resources. Various objectives of a multi-objective optimization problem (MOOP) often conflict and/or compete with one another. Complex relationships in the decision space and/or function space, such as non-convex and disconnected Pareto-optimal fronts, are also involved. In multi-criterion decision making, no single solution can be termed as the optimum solution to the multiple conflicting objectives, as a MOOP is amenable to a number of trade-off optimal solutions (Deb 2001). For this purpose, multi-objective optimization generates a Pareto front, which is a set of non-dominated solutions for problems with more

*Corresponding author. Email: nagesh@civil.iisc.ernet.in

Engineering Optimization
ISSN 0305-215X print/ISSN 1029-0273 online © 2007 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/03052150600930493

than one objective. The major goal of any MOOP algorithm is to generate a well-distributed true Pareto-optimal front or surface. A brief description of multi-objective optimization concepts is presented in the next section.

2. Multi-objective optimization

A general formulation for a multi-objective optimization problem is

$$\text{Minimize } f(x) = \{f_1(x), f_2(x), \ldots, f_m(x)\}, \quad x \in D \tag{1}$$

where $f(x)$ represents the vector of objectives and $f_i (i = 1, 2, \ldots, m)$ is a scalar decision variable which maps decision variable x into the objective space $f_i = \boldsymbol{R^n} \rightarrow \boldsymbol{R}$. The n-dimensional variable x is constrained to lie in a feasible region D which is constrained by J-inequality and K-equality constraints, i.e.

$$D = \{x : g_j(x) \leq 0, \;\; h_k(x) = 0, \quad j = 1, 2, \ldots, J; \quad k = 1, 2, \ldots, K\}. \tag{2}$$

In MOOPs, the desired goals often conflict with each other and it is not possible to satisfy them all simultaneously. Hence the Pareto-optimal solution is a solution, around which there is no way of improving any objective without degrading at least one other objective (Deb 2001). The other definitions related to Pareto solutions are given below.

- *Pareto front*. A Pareto front is a set of non-dominated solutions, being chosen as optimal, if no objective can be improved without sacrificing at least one other objective. On the other hand a solution x^* is referred to as dominated by another solution x if, and only if, x is equally good or better than x^* with respect to all objectives.
- *Weakly Pareto solution*. A solution $x^* \in D$ is said to be weakly Pareto optimal if there does not exist another solution $x \in D$ such that $f_i(x) \leq f_i(x^*)$ for all $i = 1, 2, \ldots, m$ with strict inequality for at least one i.
- *Strongly Pareto solution*. A solution $x^* \in D$ is said to be strongly Pareto optimal if there does not exist another solution $x \in D$ such that $f_i(x) < f_i(x^*)$ for all $i = 1, 2, \ldots, m$. Strongly Pareto solutions are a subset of weakly Pareto solutions. A solution which is not a weakly Pareto solution is an inferior solution.

Classical optimization problems often fail to yield true Pareto-optimal solutions when the objective function is non-convex and consists of disconnected Pareto fronts. They also require human expertise and a large number of simulation runs in order to obtain the trade-off solutions. Recently it has been emphasized that meta-heuristic techniques such as evolutionary algorithms (EAs) are attractive alternatives for solving MOOPs. Since EAs are population-based stochastic search algorithms, while solving MOOPs they can locate multiple Pareto-optimal solutions in a single run. They can also easily handle non-convex disconnected Pareto-optimal fronts. For this reason, more research is in progress on them at present and various methodologies are being evolved to solve MOOPs. Among existing multi-objective evolutionary algorithms (MOEAs), the strength Pareto EA (SPEA) (Zitzler and Thiele 1999), the Pareto-archived evolutionary strategy (PAES) (Knowles and Corne 2000), and the non-dominated sorting genetic algorithm (NSGA-II) (Deb *et al.* 2002) have been successfully used to solve MOOPs. Achieving a well-spread and well-diverse Pareto solution front is the primary goal of the MOOP. In MOEAs, apart from finding the non-dominated solution in each generation, more computational effort is required for diversity-preserving mechanisms. This computational complexity is directly related to the level of diversity and distribution that a particular MOEA aims to obtain (Deb 2001).

Another population-based meta-heuristic optimization technique, particle swarm optimization (PSO), has been applied to single-objective optimization tasks and has been found to be fast and reliable, often converging to global optimal solutions within a few steps (Kennedy and Eberhart 2001). The high speed of convergence of the PSO algorithm has led some researchers to develop multi-objective optimization algorithms using this technique. Recently, procedures for multi-objective particle swarm optimization (MOPSO) have been developed (Coello *et al.* 2004). However, it has been observed that MOPSO may encounter difficulties when solving complex problems because of its limited operators.

In this article a novel strategic procedure for MOPSO is presented and the efficiency of PSO for multiple-objective optimization is explored. Previous works on MOEAs have shown that NSGA-II performs better than PAES and SPEA (Deb *et al.* 2002). Therefore the efficiency of the proposed algorithm is demonstrated by comparison with NSGA-II and evaluated with standard performance measures taken from the MOEA literature.

The remainder of the article is organized as follows. First, a brief description of PSO is presented and the working of MOPSO is explained. The next section gives details of performance measures and test problems, and presents simulation results for unconstrained optimization for three engineering design problems. Finally, brief conclusions are outlined.

3. Particle swarm optimization

Swarm intelligence is a new area of research in that the PSO technique is inspired by studies of the social behavior of insects and animals (Kennedy and Eberhart 2001). In the PSO technique, such social behavior is modelled as optimization algorithm which guides a population of particles (the swarm) moving towards the most promising area of the search space. In PSO, each particle represents a potential solution and its position is changed according to its own experience and that of its neighbours. If the search space is D-dimensional, the ith individual (particle) of the population (swarm) can be represented by a D-dimensional vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})^{\mathrm{T}}$. The *velocity* (position change) of this particle can be represented by another D-dimensional vector $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})^{\mathrm{T}}$. The best previously visited position of the ith particle is denoted by $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})^{\mathrm{T}}$ and acts as a local guide to the particles. The swarm is manipulated according to the following two equations:

$$v_{id}^{n+1} = \chi \left[w v_{id}^{n} + c_1 r_1^{n} \frac{(p_{id}^{n} - x_{id}^{n})}{\Delta t} + c_2 r_2^{n} \frac{(p_{gd}^{n} - x_{id}^{n})}{\Delta t} \right] \tag{3}$$

$$x_{id}^{n+1} = x_{id}^{n} + \Delta t \; v_{id}^{n+1} \tag{4}$$

where g is the index of the global guide of a particle, the superscripts denote the iteration number $d = 1, 2, \ldots, D, i = 1, 2, \ldots, N$, N is the size of the swarm population, χ is a constriction factor which controls and constricts the magnitude of the velocity, w is the inertia weight which is often used as a parameter to control exploration and exploitation in the search space, c_1 and c_2 are positive constant parameters called acceleration coefficients, r_1 and r_2 are random numbers uniformly distributed in [0, 1], Δt is the time step, usually set as 1, and n is the iteration number.

The similarities between PSO and the EAs allow the algorithm to be extended to handle multiple objectives. For example, PSO maintains a population of solutions, which allows simultaneous exploration of different parts of the Pareto front. By incorporating the Pareto dominance principle into the PSO algorithm, MOPSO techniques are proposed. However, the main difficulties in extending PSO to multi-objective problems are finding the best way of

selecting the guides for each particle in the swarm and finding an effective strategy to maintain diversity in the population and to achieve convergence to true Pareto-optimal solutions. The difficulty is apparent as there are no clear concepts of personal and global bests that can be clearly identified when dealing with many objectives rather than a single objective. In addition, the diversity maintenance mechanism while seeking global Pareto-optimal solutions is rather poor compared with other techniques such as MOEAs.

Some approaches to extend the PSO technique to multi-objective optimization have recently been reported. Ray and Liew (2002) proposed a swarm metaphor approach, which uses the Pareto dominance relation and combines concepts of evolutionary techniques with the particle swarm. Parsopoulos and Vrahatis (2002) proposed a MOPSO algorithm, adopting different types of aggregating functions to solve MOOPs. Hu and Eberhart (2002) proposed a dynamic neighbourhood PSO, in which only one objective is optimized at a time using a scheme similar to lexicographic ordering. A revised version of this approach, which uses a secondary population, was presented by Hu *et al.* (2003). Coello and Lechuga (2002) proposed a MOPSO based on the idea of having a global repository in which every particle deposits its flight experiences after each flight cycle. The repository is updated using a geographically based system defined in terms of the objective function values of each individual. This repository is used by the particles to identify a leader that will guide the search. A revised version of the approach, in which a special mutation operation on decision variable space is used to improve the performance of the algorithm, was presented by Coello *et al.* (2004). Fieldsend and Singh (2002) incorporated an unconstrained elite archive to store the non-dominated individuals found during the search process. The archive interacts with the primary population in order to define local guides. This approach also uses a mutation operator which acts on the velocity value used by PSO. Li (2003) proposed a non-dominated sorting PSO, which incorporates the main mechanisms of NSGA-II (Deb *et al.* 2002) into a PSO algorithm. This approach showed a very competitive performance when compared with NSGA-II.

Most of these studies basically adopted procedures from the MOEA literature. At present much research on MOPSO is under way, with particular attention being paid to reducing the complexity of computations to speed up the model performance, identifying an efficient way of selecting the global guides; and creating effective selection pressure for reaching the true Pareto-optimal region. In this study an efficient procedure for MOPSO called elitist-mutated MOPSO (EM-MOPSO) is explored. Details of the algorithm are given in the next section.

4. Multi-objective particle swarm optimization procedure

The procedure combines Pareto-dominance principles with PSO and uses elitism in its evolution. The main algorithm consists of initialization of the population, evaluation, performing PSO operations, and reiterating the search on the swarm to reach true Pareto-optimal solutions. In this process, the particles are first evaluated and checked for dominance relations among the swarm. Then the non-dominated solutions found are stored in an external repository (*ERP*). The size of this repository is restricted to a predefined number. This restriction is imposed using a crowding distance comparison operator (Deb *et al.* 2002), which gives the density measure of the existing particles in the function space. The *ERP* with the crowding operator helps the particles to create effective selection pressure toward true Pareto-optimal solutions. A stepwise linearly variable *ERP* is used in this procedure. This helps to achieve a well-distributed Pareto front and saves considerable computational time during optimization. The selection of global guides for each particle is performed by randomly choosing one solution from those stored in the *ERP*. In addition, an efficient strategic mechanism

called elitist-mutation is incorporated into the algorithm. By attracting the swarm towards sparsely populated regions in the *ERP*, it helps the search to distribute the non-dominated solutions uniformly along the true Pareto-optimal front. Some of the main mechanisms used in this procedure are explained below.

4.1 *Elitist-mutation*

In this study a new strategic mutation mechanism called elitist-mutation is proposed to improve the performance of the algorithm. The idea underlying this method is to effectively explore and exploit the search space in the feasible region, where members of the repository that are isolated in the non-dominated *ERP* should be preferentially mutated and replace the poorest particles in the swarm. This mechanism acts on a predefined number of particles. In the initial phase, it replaces the infeasible solutions with the least crowded solutions in *ERP*, after performing the elitist mutation mechanism on them, and in the later phase it tries to exploit the search space around the sparsely populated particles along the Pareto fronts.

This is a special mechanism which helps to overcome the drawbacks of the traditional PSO algorithm when it is extended to MOOPs. It also promotes diversity in the population and consequently helps the search to find the true Pareto-optimal front. The pseudo-code of the elitist-mutation mechanism is given below:

(i) Randomly select one of the objectives from m objectives. Sort the fitness function in descending order and obtain the index numbers (*DscSortPos*) for the respective particles.
(ii) Use the crowding distance assignment operator to calculate the density of solutions in the *ERP* and sort them in descending order of crowding value. Randomly select one of the sparsely populated solutions from the top 10% of the *ERP* as a guide (g).
(iii) Perform an elitist-mutation on a predefined number of particles (*nMutMax*).

Let Rp be the length of repository $|ERP|$, p_{em} be the probability of mutation, *mutScale* be mutation scale used to preserve diversity, $intRandom(a, b)$ be an integer random number in the interval $[a, b]$, *rand* be a uniformly distributed random number $U(0, 1)$, *randn* be a Gaussian random number $N(0, 1)$, and $VR[i]$ be a range of decision variable i.

For $i = 1$ to *nMutMax*
    $l = DscSortPos(i)$
    $g = intRandom(1,\ 0.1 * Rp)$
    For $d = 1$ to dim
        if $(rand < p_{em})$
            $X[1][d] = ERP[g][d] + mutScale * VR[d] * randn$
        else
            $X[1][d] = ERP[g][d]$.

If the mutated value exceeds the bounds, then it is limited to the upper or lower bound. During this elitist-mutation step, the velocity vector of the particle is unchanged.

4.2 *Crowding distance computation procedure*

The crowding distance value of a solution provides an estimate of the density of solutions surrounding that solution (Deb *et al.* 2002). Crowding distance is calculated by first sorting the set of solutions in ascending order of objective function values. The crowding distance value of a particular solution is the average distance of its two neighbouring solutions. The boundary

solutions which have the lowest and highest objective function values are given infinite crowding distance values, so that they are always selected. This process is done for each objective function. The final crowding distance value of a solution is computed by adding all the individual crowding distance values in each objective function. An efficient quick sorting procedure is used. The pseudo-code of the crowding distance computation is given below.

(i) Get the number of non-dominated solutions in the external repository (*ERP*)
$l = |ERP|$
(ii) Initialize distance
For $i = 1$ to l
$ERP[i]\cdot\text{distance} = 0$
(iii) Compute the crowding distance of each solution
For each objective m
Sort using objective value
$ERP = \text{sort}(ERP, m)$
Set the boundary points to a large value, so that they are always selected
$ERP[1]\cdot\text{distance} = ERP[l]\cdot\text{distance} = \infty$
For $i = 2$ to $(l - 1)$
$ERP[i]\cdot\text{distance} = ERP[i]\cdot\text{distance} + (ERP[i+1]\cdot m - ER[i-1]\cdot m)/(f_m^{\max} - f_m^{\min})$

4.3 *Handling overlapping solutions*

When elitism is used in multi-objective optimization, some overlapping solutions may exist in the population or the external repository. In such a case an effective removal strategy is performed in the objective space. Only a single solution among the overlapping solutions with the same objective vector is left in the current population, i.e. overlapping solutions are removed, so that each solution in the current population has a different location in the objective space. This also enhances the performance of the algorithm.

4.4 *EM-MOPSO algorithm*

The EM-MOPSO algorithm obtained using the above mechanisms can be summarized as follows.

Step 1 Initialize the population position and velocity vectors. The current position X_i of the ith particle is initialized with random real numbers within the specified decision variable range; each particle velocity vector V_i is initialized with a uniformly distributed random number in [0,1].

Step 2 Evaluate each particle in the population. The personal best position P_i is set to X_i. Identify particles that give non-dominated solutions in the current population and store them in an external repository (*ERP*). Set the iteration Counter, t to 0.

Step 3 Randomly select a global best P_g for the ith particle from the solutions stored in the *ERP*. Calculate the new velocity V_i, and the new X_i using equations (3) and (4), respectively. Repeat the procedure for all the particles.

Step 4 Evaluate each particle in the population. Then check each individual for dominance with its current personal best solution. Replace P_i with the current solution if the new one dominates the current P_i.

Step 5 Set *ERP* to a temporary repository (*TempERP*) and empty *ERP*. Identify particles that give non-dominated solutions in the current iteration and add them to *TempERP*.

Step 6 Find the non-dominated solutions in *TempERP*. If the number of non-dominated solutions found exceeds the desired size of the *ERP*, use the crowding distance operator to select the desired solutions and store them in the *ERP*. Empty the *TempERP*.

Step 7 Re-sort the *ERP* according to crowding distance values and perform the elitist-mutation operation on specified number of particles.

Step 8 Increment the iteration Counter, t to $t + 1$ and check for termination criteria. If the termination criterion is not satisfied, go to step 3; otherwise output the non-dominated solution set from *ERP*.

The algorithm was implemented following these steps and tested on various problems as discussed in the next section.

5. Experiments and results

The EM-MOPSO algorithm has been applied to several test problems taken from the literature, including both unconstrained and constrained optimization problems. Standard performance measures of MOEAs have been used to evaluate the performance of the proposed algorithm. A brief description of the performance measures is given below.

5.1 *Performance measures*

For any multi-objective problem, the main goals of optimization are to minimize the divergence of the Pareto front produced by an algorithm from the true Pareto front (i.e. convergence to the true Pareto-optimal solution set) and to maximize the spread of solutions found (i.e. maintain diversity among the generated set of solutions). This should also ensure that the solutions generated are uniformly distributed along the true Pareto-optimal front. Various performance measures for evaluating a set of non-dominated solutions have been proposed in the literature. Three metrics for the test problems have been used to test the performance of the algorithm developed in this study. Following Deb *et al.* (2002), the average performance is calculated for each test problem over 20 runs. However, it should be noted that no performance measure can simultaneously evaluate various aspects of a solution set (Knowles and Corne 2000, Zitzler *et al.* 2000). Therefore a sample solution from a single run is shown graphically for each of the test problems with two objective functions.

5.1.1 Set coverage metric. This metric was proposed by Zitzler (1999) and gives the relative spread of solutions between two sets of solution vectors A and B. The set coverage metric calculates the proportion of solutions in B, which are weakly dominated by solutions of A:

$$C(A, B) = \frac{|\{b \in B | \exists a \in A : a \leq b\}|}{|B|}. \tag{5}$$

If $C(A, B) = 1$ all solutions in B are weakly dominated by A, and if $C(A, B) = 0$ none of the solutions in B are weakly dominated by A.

5.1.2 Generational distance. This metric gives the closeness of the Pareto-optimal solutions obtained to the true Pareto-optimal solutions. Let Q be a solution set obtained by a MOEA. The proximity of Q to the Pareto front is evaluated by the generational distance (GD)

defined as follows (Veldhuizen 1999):

$$GD = \frac{\left(\sum_{i=1}^{|Q|} d_i^p\right)^{1/p}}{|Q|} \tag{6}$$

where P^* is a reference solution set (i.e. the set of all possible true Pareto-optimal solutions).

A Euclidian distance-based metric is used in this study. Therefore, for $p = 2$, the parameter d_i is the Euclidean distance between the solution obtained by the algorithm $i \in Q$ and the nearest member of true Pareto-optimal solutions P^*:

$$d_i = \min_{k=1}^{|P^*|} \sqrt{\sum (f_m^{(i)} - f_m^{*(k)})^2} \tag{7}$$

where $f_m^{*(k)}$ is the mth objective function value of the kth member of P^*.

5.1.3 Spread. The spread or diversity metric (Δ) measures the extent of spread achieved among the solutions obtained. Here the main interest is finding a set of solutions which span the entire Pareto-optimal region. Deb *et al.* (2002) suggested the following metric for effective measurement of the spread:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{\sum_{m=1}^{M} d_m^e + (N-1)\bar{d}} \tag{8}$$

where d_i is the Euclidean distance between consecutive solutions in the non-dominated set of solutions obtained, $\bar{d}$ is the average of all distances d_i, $i = 1, 2, \ldots, (N-1)$, assuming that there are N solutions on the best non-dominated front, and d_m^{e} is the distance between the extreme solutions of true Pareto-optimal solutions and the Pareto-optimal solutions obtained corresponding to the mth objective function.

5.2 *Unconstrained optimization*

The details of the test problems for unconstrained optimization are given in table 1. The first and second problems (SCH and FON) were suggested by Schaffer (1987) and Fonseca and Fleming (1998), respectively, and the remainder (ZDT problems) were suggested by Zitzler *et al.* (2000). The sensitivity of PSO parameters for single objective optimization was thoroughly investigated in an earlier study (Kumar and Reddy 2006). However, in this study a thorough sensitivity analysis was also carried out for various parameters of EM-MOPSO algorithm, after which the following parameters were adopted: initial population, 100; constants c_1 and c_2, 1.0 and 0.5, respectively; inertial weight w, 1; constriction coefficient χ, 0.9; number of non-dominated solutions to be found (*maxERP*), 100. In the elitist-mutation step, the size of the elitist-mutated particles was set at 15, p_{em} was set at 0.2, and the value of *mutScale* decreased from 0.2 to 0.01 over the iterations. In order to avoid any ambiguity in the performance of the algorithm with respect to the number of iterations, a sufficient number of iterations was chosen depending on the complexity of the problem. For test problems SCH and FON, EM-MOPSO was run for 250 iterations, whereas 500 iterations were used for ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6.

For NSGA-II, the initial population size was 100, the crossover probability was 0.9, and the mutation probability was $1/n$ (n is the number of real variables). The SBX and real parameter mutation were 15 and 20, respectively (all these parameter values are similar to those used by Deb *et al.* (2002)). The number of iterations was the same as used for EM-MOPSO.

Table 1. Test problems used for the performance evaluation of EM-MOPSO.

Test problem	n	Variable bounds	Objective functions
SCH	1	$[-10^3, 10^3]$	$f_1(x) = x^2$ $f_2(x) = (x-2)^2$
FON	3	$[-4, 4]$	$f_1(x) = 1 - \exp\left[-\sum_{i=1}^{3}\left(x_i - \frac{1}{\sqrt{3}}\right)^2\right]$ $f_1(x) = 1 - \exp\left[-\sum_{i=1}^{3}\left(x_i - \frac{1}{\sqrt{3}}\right)^2\right]$
ZDT1	30	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)\{1 - \sqrt{[x_1/g(x)]}\}$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$
ZDT2	30	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)\{1 - [x_1/g(x)]^2\}$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$
ZDT3	30	$[0, 1]$	$f_1(x) = x_1$ $f_2(x) = g(x)\{1 - \sqrt{[x_1/g(x)]} - [x_1/g(x)]\sin(10\pi x_1)\}$ $g(x) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$
ZDT4	10	$x_1 \in [0, 1]$ $x_i \in [-5, 5], i = 2, \ldots, n$	$f_1(x) = x_1$ $f_2(x) = g(x)\{1 - \sqrt{[x_1/g(x)]}\}$ $g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}[x_i^2 - 10\cos(4\pi x_i)]$
ZDT6	10	$[0, 1]$	$f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(x) = g(x)\{1 - [f_1(x)/g(x)]^2\}$ $g(x) = 1 + 9\left[\left(\sum_{i=2}^{n} x_i\right)/(n-1)\right]^{0.25}$

The EM-MOPSO algorithm was applied to the above set of test problems. Some of these are by far the most difficult problems for unconstrained multi-objective optimization suggested in the literature (Deb *et al.* 2002). Table 2 compares the performances of EM-MOPSO and NSGA-II, showing the best, worst, mean, variance, and standard deviation (SD) values for the performance measures considered in the study (i.e. set coverage metric, generational distance, and spread metric). The statistics shown are based on 20 random trials. It should be noted that the results are compared based on model performance in a run. In all test problems, a set of $|P^*| = 500$ uniformly spaced true Pareto-optimal solutions were used to calculate SC and GD metrics. In the set coverage metric $SC(P^*, A)$, P^* is the true Pareto-optimal solution and A is the non-dominated solution set obtained from a MOOP algorithm by either EM-MOPSO or NSGA-II.

It can be seen from table 2 that EM-MOPSO is able to converge closely to the true Pareto-optimal solution, with good distribution of non-dominated solutions, for most of the problems. On the basis of SC metric, EM-MOPSO performed better than NSGA II as five (SCH, FON, ZDT1, ZDT4, ZDT6) of seven test cases have minimal mean SC values compared with NSGA-II, whereas NSGA-II has the minimal mean value for only two test cases (ZDT2, ZDT3). Table 2 also shows the GD statistics for 20 simulation runs for both the algorithms. This GD metric gives an idea of how closely a solution set converges to true Pareto-optimal solutions. Again, EM-MOPSO performs well for five test cases (SCH, FON, ZDT1, ZDT4, ZDT6), while NSGA-II performance is good for the other two cases. The diversity of non-dominated solutions is assessed by the spread metric Δ for both EM-MOPSO and NSGA-II simulations: a lower value of Δ indicates a better performance. Table 2 shows that EM-MOPSO is able to

Table 2. Performance measures of EM-MOPSO and NSGA-II for test problems considered in the study showing the best, worst, mean, variance, and standard deviation (SD) values for set coverage metric (SC), generational distance (GD), and spread metric (Δ).*

		SC		GD		Δ	
Test case	Statistic	EM-MOPSO	NSGA-II	EM-MOPSO	NSGA-II	EM-MOPSO	NSGA-II
SCH	Best	0.00000	0.00000	0.00865	0.00843	0.31626	0.31686
	Worst	0.02000	0.12000	0.01034	0.01082	0.40266	0.41266
	Mean	**0.00684**	0.04778	**0.00949**	0.00983	**0.35363**	0.35462
	Variance	0.00003	0.00212	0.00000	0.00000	0.00064	0.00066
	SD	0.00584	0.04604	0.00047	0.00075	0.02537	0.02567
FON	Best	0.73000	0.89000	0.00471	0.00578	0.21432	0.32664
	Worst	0.83000	0.99000	0.00533	0.00645	0.29232	0.39060
	Mean	**0.77316**	0.93667	**0.00505**	0.00611	**0.24929**	0.36685
	Variance	0.00072	0.00087	0.00000	0.00000	0.00055	0.00038
	SD	0.02685	0.02958	0.00017	0.00023	0.02347	0.01956
ZDT1	Best	0.07000	0.08000	0.00466	0.00365	0.22189	0.31545
	Worst	0.32000	0.28000	0.00547	0.00608	0.27169	0.42042
	Mean	**0.18240**	0.19550	**0.00513**	0.00524	**0.24502**	0.35750
	Variance	0.00379	0.00345	0.00000	0.00000	0.00037	0.00083
	SD	0.06156	0.05876	0.00012	0.00014	0.01923	0.02889
ZDT2	Best	0.13000	0.07000	0.00396	0.00312	0.25101	0.32387
	Worst	0.42000	0.20000	0.00548	0.00376	0.32698	0.39442
	Mean	0.22100	**0.13000**	0.00459	**0.00332**	**0.28977**	0.35368
	Variance	0.00302	0.00135	0.00000	0.00000	0.00048	0.00062
	SD	0.05495	0.03674	0.00039	0.00023	0.02181	0.02482
ZDT3	Best	0.15000	0.11000	0.00675	0.00655	0.69634	0.61442
	Worst	0.49000	0.21000	0.00837	0.00711	0.85543	0.64894
	Mean	0.33450	**0.16333**	0.00720	**0.00677**	0.76013	**0.62978**
	Variance	0.00601	0.00142	0.00000	0.00000	0.00174	0.00015
	SD	0.07752	0.03775	0.00038	0.00019	0.04174	0.01239
ZDT4	Best	0.00000	0.13000	0.00315	0.00335	0.30930	0.28178
	Worst	0.34000	0.94000	0.00595	0.00505	0.43612	0.48404
	Mean	**0.07350**	0.50333	**0.00379**	0.00406	**0.35393**	0.36810
	Variance	0.00983	0.09175	0.00000	0.00000	0.00113	0.00191
	SD	0.09917	0.30290	0.00087	0.00062	0.03364	0.04371
ZDT6	Best	0.00000	0.24000	0.00521	0.00621	0.38126	0.59876
	Worst	0.03000	0.37000	0.01145	0.00667	0.88011	0.65439
	Mean	**0.00500**	0.29889	**0.00632**	0.00650	**0.53392**	0.61615
	Variance	0.00008	0.00169	0.00000	0.00000	0.01460	0.00043
	SD	0.00889	0.04106	0.00132	0.00016	0.12085	0.02062

SC, Set coverage metric (P^*, A) where P^* is the true Pareto-optimal solution set and A is the algorithm of interest (EM-MOPSO or NSGA-II).
*The results are based on 20 random trials for both algorithms.

find a better spread of solutions than NSGA-II for most of the test cases (SCH, FON, ZDT1, ZDT2, ZDT4 and ZDT6); NSGA-II performs better only for ZDT3.

In order to demonstrate the working of the algorithm, a typical simulation run using EM-MOPSO and NSGA is shown for all these test problems. The Pareto-optimal solution sets obtained for SCH and FON are shown in figures 1 and 2 where it can be seen that both the algorithms achieve true Pareto-optimal fronts. Figures 3 and 4 show all the non-dominated solutions for ZDT1 and ZDT2, respectively, obtained after 500 iterations with EM-MOPSO and NSGA-II. It can be seen that ZDT1 has a convex objective space, whereas ZDT2 has a non-convex Pareto-optimal front. These figures demonstrate the abilities of EM-MOPSO to converge to the true Pareto-optimal front and to find a diverse set of solutions in this front. Figure 5 shows the non-dominated solutions for the ZDT3 problem, which has disconnected

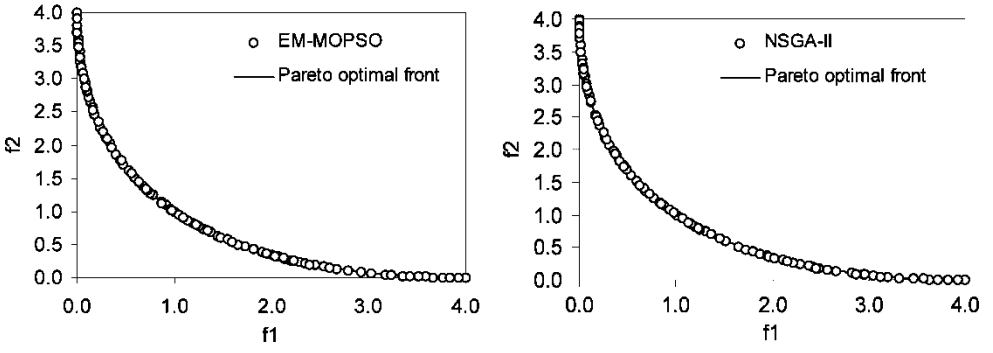

Figure 1. Non-dominated solutions obtained with EM-MOPSO and NSGA-II for problem SCH.

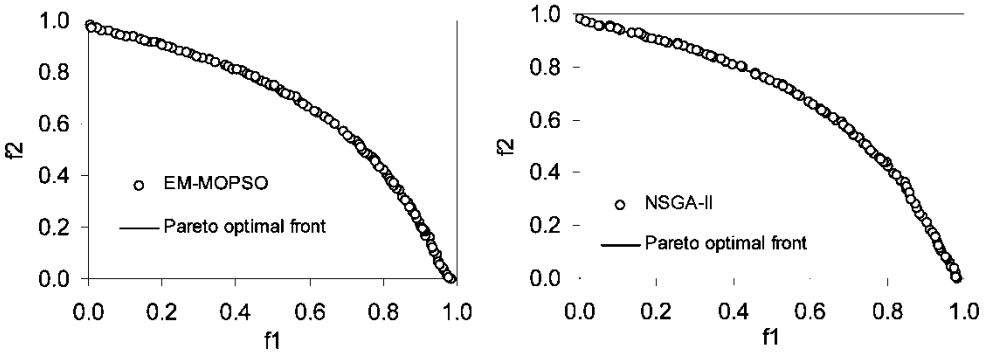

Figure 2. Non-dominated solutions obtained with EM-MOPSO and NSGA-II for problem FON.

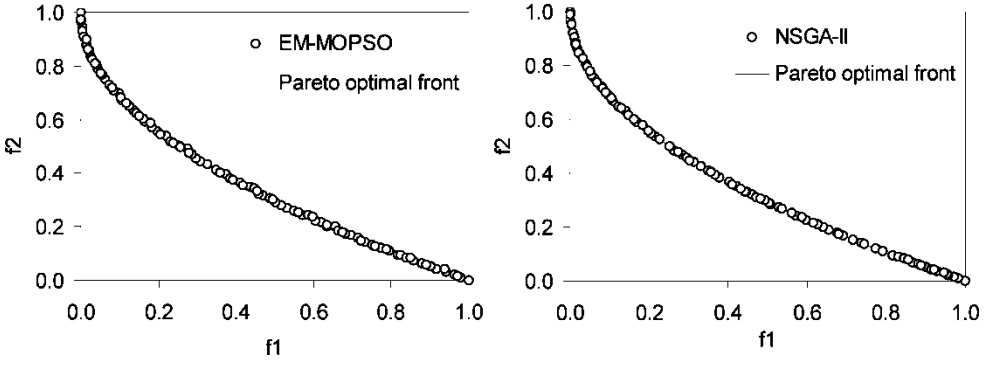

Figure 3. Non-dominated solutions obtained with EM-MOPSO and NSGA-II for problem ZDT1.

Pareto fronts. It can be clearly seen that the algorithm captures those disconnected fronts quite well. Test problem ZDT4 has 21^9 different local Pareto-optimal fronts in the search space, of which only one corresponds to the true Pareto-optimal front (Deb *et al.* 2002). This is one of the classic MOOPs. Figure 6 shows that EM-MOPSO is able to achieve a near-global Pareto-optimal front, again proving its efficiency. Figure 7 shows that EM-MOPSO finds a uniformly distributed set of non-dominated solutions to problem ZDT6 which converges better than the solution obtained by NSGA-II. The graphical illustration also clearly demonstrates that the proposed algorithm is efficient and is able to achieve true Pareto-optimal solutions.

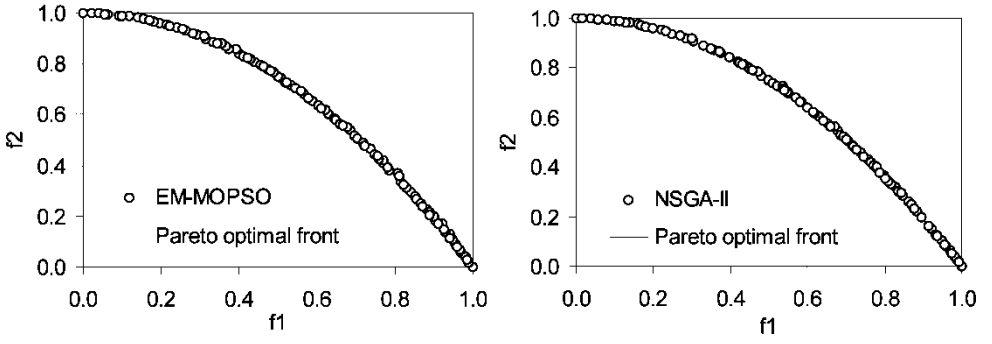

Figure 4. Non-dominated solutions obtained with EM-MOPSO and NSGA-II for problem ZDT2.

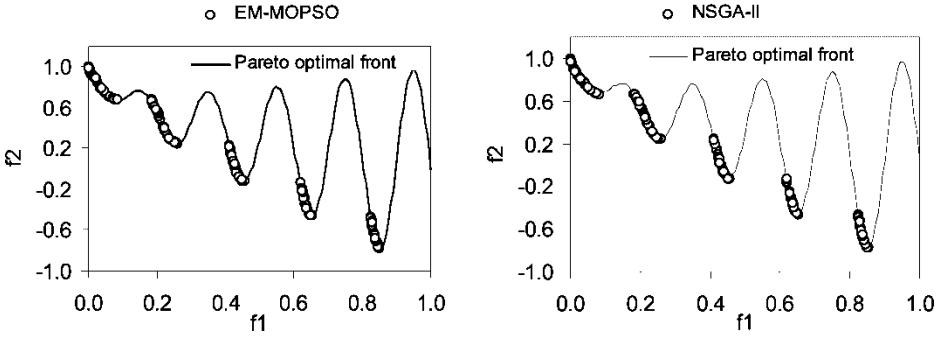

Figure 5. Non-dominated solutions obtained with EM-MOPSO and NSGA-II for problem ZDT3.

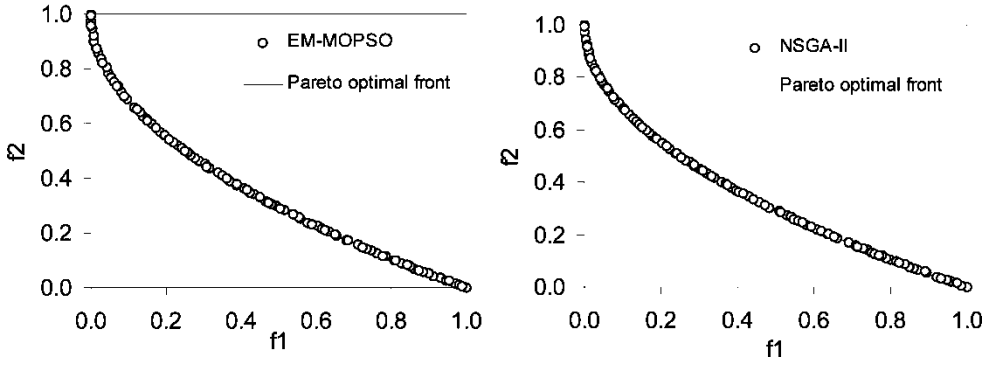

Figure 6. Non-dominated solutions obtained with EM-MOPSO and NSGA-II for problem ZDT4.

5.3 *Constrained optimization*

The EM-MOPSO algorithm has also been tested for constrained optimization for some engineering design problems. In general, when comparing two feasible particles, the particle which dominates the other particle is considered to be a better solution. However, if both particles are infeasible, the particle with fewer constraint violations is the better solution.

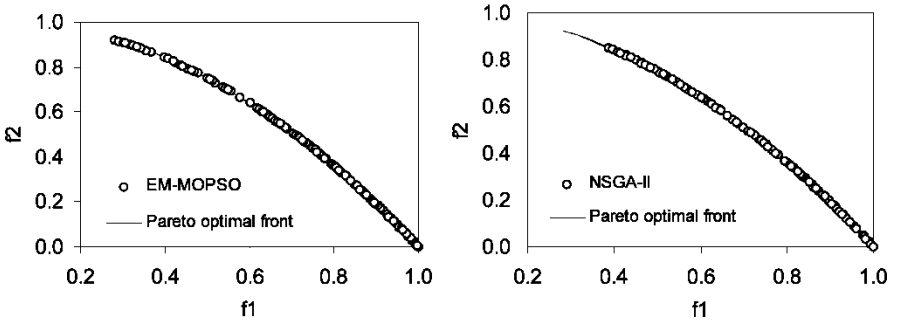

Figure 7. Non-dominated solutions obtained with EM-MOPSO and NSGA-II for problem ZDT6.

The mechanism used by Deb *et al.* (2002) for handling constrained optimization problems was adopted because of its simplicity in using feasibility and non-dominance of solutions when comparing solutions. A solution i is said to constrained-dominate a solution j if any of the following conditions are satisified.

1. Solution i is feasible and solution j is infeasible.
2. Both solution i and solution j are infeasible, but solution i has a smaller overall constraint violation.
3. Both solution i and solution j are feasible and solution i dominates solution j.

The following parameters were used in the application of EM-MOPSO: population size, 100; constant parameters c_1 and c_2, 1.0 and 0.5, respectively; inertial weight w, 1; constriction coefficient χ, 0.9; size of external repository (*maxERP*), 100. The value of p_{em} was set at 0.2, and *mutScale* decreases from 0.2 to 0.01 over the iterations. The parameters used for NSGA-II were as follows: population size, 100; crossover probability, 0.9; mutation probability, $1/n$ (n is the number of real variables). The distribution indices for real-coded crossover and mutation operators are set to 20 and 100, respectively. The maximum number of iterations was set at 100 for both algorithms. The same parameter settings were used for all the problems. The consistency of the algorithm was verified by running all the problems for several trial runs. However, as the performance has already been proved (see previous section), only the results for the best sample run in 20 trial runs are reported here.

5.3.1 Two-bar truss design. This problem was originally studied using the ε-constraint method (Palli *et al.* 1999). Later it was studied using NSGA-II (Deb *et al.* 2000). As shown in figure 8, the truss has to carry a certain load without elastic failure. Thus, in addition to the objective of designing the truss for minimum volume (which is equivalent to designing

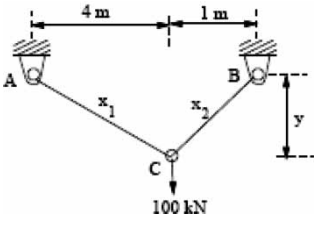

Figure 8. The two-bar truss design problem (Deb *et al.* 2000). The two objectives are minimization of the cost of fabrication and of the stresses in members (AC and BC).

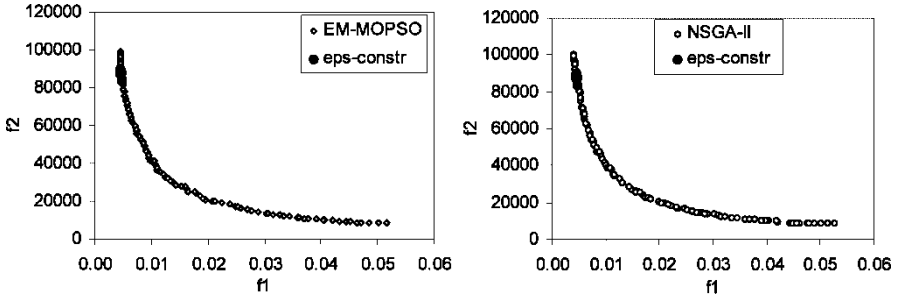

Figure 9. Non-dominated solutions obtained using EM-MOPSO and NSGA-II for the two-bar truss design problem. The optimal solutions found using the ε-constraint method (eps-constr) are also shown for comparison.

for minimum cost of fabrication), there are additional objectives of minimizing the stresses in each of the two members AC and BC. Therefore the design problem involves a two-objective optimization problem for three variables: vertical distance y between B and C (metres), length x_1 of AC (metres), and length x_2 of BC in (metres). The two-bar truss design problem can be expressed in mathematical form as follows:

$$\begin{aligned}
&\text{minimize} \quad f_1(x) = x_1\sqrt{16 + y^2} + x_2\sqrt{1 + y^2} \\
&\text{minimize} \quad f_2(x) = \max(\sigma_{AC}, \sigma_{BC}) \\
&\text{subject to} \\
&\qquad \max(\sigma_{AC}, \sigma_{BC}) \geq 1(10)^5 \\
&\qquad 1 \leq y \leq 3 \quad \text{and} \quad x \geq 0.
\end{aligned}$$

The stresses are calculated as follows:

$$\sigma_{AC} = \frac{20\sqrt{16 + y^2}}{yx_1} \quad \sigma_{BC} = \frac{80\sqrt{1 + y^2}}{yx_2}.$$

To apply the proposed method, the bounds on x_i are taken as $0 \leq x_i \leq 0.01$ for $i = 1, 2$. The ε-constraint method reported only five solutions with the following spread: (0.004445 m^3, 89983 kPa) and (0.004833 m^3, 83268 kPa). Figure 9 shows the optimized fronts obtained using the ε-constraint, EM-MOPSO, and NSGA-II methods. The solutions obtained with EM-MOPSO are spread in the range (0.004026 m^3, 99996 kPa) and (0.05273 m^3, 8434.493 kPa), and those obtained with NSGA-II are spread in the range (0.00407 m^3, 99755 kPa) and (0.05304 m^3, 8439 kPa). Thus both have a wide variety of alternatives. However, the ε-constraint method was unable to find much variety in the solutions in terms of the second objective (Palli *et al.* 1999). If minimum volume is desired, EM-MOPSO gives a value as low as 0.004026 m^3. If minimization of stress is important, it finds a solution with stress as low as 8434.493 kPa, whereas the ε-constraint method finds a solution with a minimumstress of 83268 kPa, which is nearly 10 times higher. The EM-MOPSO solutions are very competitive with NSGA-II solutions in terms of both closeness to the true optimum front and their spread.

5.3.2 I-beam design. The second design problem is taken from Yang *et al.* (2002). The problem is to find the dimensions of the beam shown in figure 10. In this design problem, it should satisfy the dimensions of the geometric and strength constraints, and at the same time

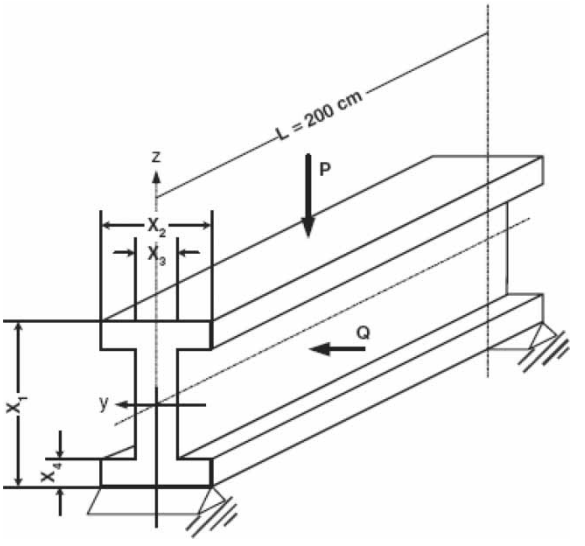

Figure 10. The I-beam design problem (Yang *et al.* 2002). The objectives are minimization of cross-sectional area of the beam and the static deflection of the beam.

minimize the cross-sectional area of the beam and the static deflection of the beam under a force P. The mathematical form of the problem is as follows:

Minimize cross-sectional area (cm^2)

$$f_1 = 2x_2x_4 + x_3(x_1 - 2x_4)$$

Minimize displacement (cm)

$$f_2 = \frac{PL^3}{48EI}$$

where

$$I = \frac{1}{12}\{x_3(x_1 - 2x_4)^3 + 2x_2x_4[4x_4^2 + 3x_1(x_1 - 2x_4)]\}.$$

Find x_i, $i = 1, 2, \ldots, 4$

subject to

$$g(x) = \sigma_a - \left(\frac{M_y}{Z_y} + \frac{M_z}{Z_z}\right) \geq 0$$

$$10 \leq x_1 \leq 80, \quad 10 \leq x_2 \leq 50, \quad 0.9 \leq x_3 \leq 5, \quad 0.9 \leq x_4 \leq 5$$

where

$$M_y = \frac{P}{2} \times \frac{L}{2}, \quad M_z = \frac{Q}{2} \times \frac{L}{2}$$

$$Z_y = \frac{1}{6x_1}\{x_3(x_1 - x_4)^3 + 2x_2x_4[4x_4^2 + 3x_1(x_1 - 2x_4)]\}$$

$$Z_z = \frac{1}{6x_2}\{(x_1 - x_4)x_3^3 + 2x_4x_2^3\}$$

$E = 2 \times 10^4\,\text{kN cm}^{-2}$, $\sigma_\text{a} = 16\,\text{kN cm}^{-2}$, $P = 600\,\text{kN}$, $Q = 50\,\text{kN}$, and $L = 200\,\text{cm}$.

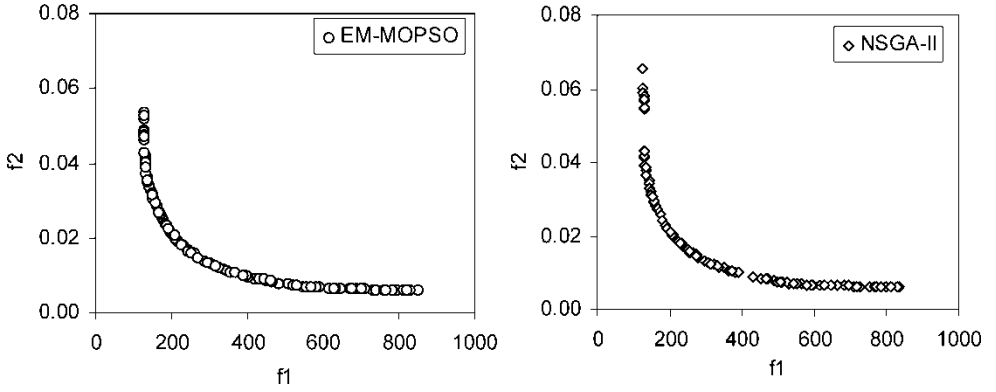

Figure 11. Non-dominated solutions obtained using EM-MOPSO and NSGA-II for the I-beam design problem.

Figure 11 shows the non-dominated solutions obtained after 100 iterations for both EM-MOPSO and NSGA-II. It can be seen that EM-MOPSO is able to maintain a uniform distribution of solutions. EM-MOPSO obtained the minimal cross-sectional area of 127.9508 units for a deflection of 0.05368 units, and for the minimal deflection of 0.005961 units the cross-sectional area is 829.5748 units. NSGA-II obtained a minimal cross-sectional area of 127.2341 units with deflection of 0.0654 units, and a minimal deflection of 0.0060 units with cross-sectional area of 829.8684 units. Thus the proposed method is able to find a wide spread of Pareto-optimal solutions.

5.3.3 Welded beam design. The third design problem was studied by Deb *et al.* (2000). A beam needs to be welded on to another beam and must carry a certain load (figure 12). The overhang has a length of 14 inches and a force F of 6000 lb is applied at the end of the beam. The objective of the design is to minimize the cost of fabrication and the end deflection. The mathematical formulation of the two objective optimization problem is as follows.

Minimize

$$f_1(x) = 1.10471\,h^2 l + 0.04811 tb(14.0 + l)$$
$$f_2(x) = \delta(x)$$

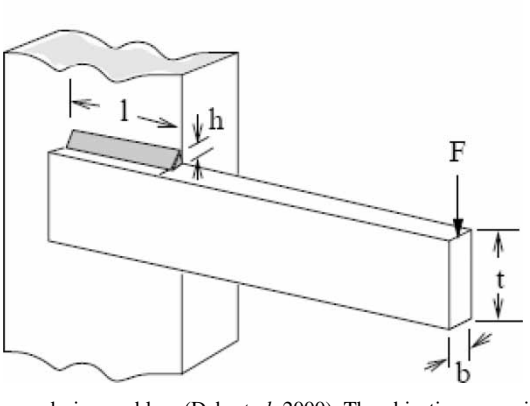

Figure 12. The welded beam design problem (Deb *et al.* 2000). The objectives are minimization of the cost and the end deflection of the beam.

subject to

$$g_1(x) = 13,600 - \tau(x) \geq 0$$
$$g_2(x) = 30,000 - \sigma(x) \geq 0$$
$$g_3(x) = b - h \geq 0$$
$$g_4(x) = P_c(x) - 6000 \geq 0.$$

The deflection term $\delta(x)$ is given by

$$\delta(x) = \frac{2.1952}{t^3 b}.$$

The first constraint ensures that the shear stress developed at the support location of the beam is less than the allowable shear strength of the material (13 600 psi). The second constraint ensures that the normal stress developed at the support location of the beam is less than the allowable yield strength of the material (30 000 psi). The third constraint ensures that the thickness of the beam is not less than the weld thickness from a practical standpoint. The fourth constraint ensures that the allowable buckling load of the beam (along the t direction)is greater than the applied load F. The stress and buckling terms are as follows:

$$\tau(x) = \sqrt{(\tau')^2 + (\tau'')^2 + \frac{(l\tau'\tau'')}{\sqrt{0.25(l^2 + (h+t)^2)}}}$$
$$\tau' = \frac{6,000}{\sqrt{2}hl}$$
$$\tau'' = \frac{6000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2\{0.707hl(l^2/12 + 0.25(h+t)^2)\}}$$
$$\sigma(x) = \frac{504,000}{t^2 b}$$
$$P_c(x) = 64,746.022(1 - 0.0282346t)tb^3.$$

The variables are initialized in the following ranges: $0.125 \leq h, b \leq 5.0$, and $0.1 \leq l, t \leq 10.0$. The EM-MOPSO algorithm is applied with the same parameters and run for 100 iterations. Figure 13 shows the optimized non-dominated solutions obtained using EM-MOPSO and NSGA-II. Both algorithms are able to find a wide variety of solutions which are uniformly

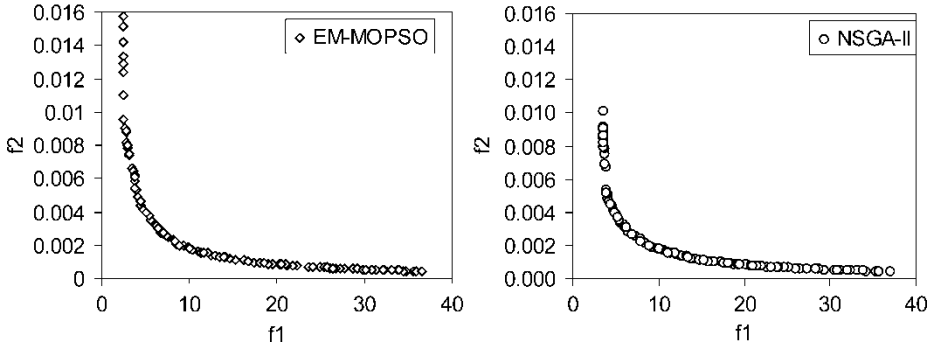

Figure 13. Optimized non-dominated solutions obtained using EM-MOPSO and NSGA-II for the welded beam design problem.

spread. EM-MOPSO found the minimal cost solution as 2.382 units with deflection 0.0157 inches, and the minimal deflection as 0.000439 with a cost of 36.4836 units. For NSGA-II, the minimal cost is 3.443 units for deflection of 0.0101 units, and the minimal deflection is 0.004 with a cost of 36.9121 units. The extreme solutions are captured well in EM-MOPSO. Again, it is demonstrated that the proposed algorithm is efficient and is able to find a wide variety of Pareto-optimal solutions.

It is more difficult to maintain a good spread of solutions in constrained optimization problems than in unconstrained problems, because in order to maintain a spread of solutions on the constraint boundary the solutions have to be modified in a particular manner as dictated by the constraint function (Deb *et al.* 2002). However, the results clearly show that the proposed algorithm does not have any difficulty in achieving a good spread of Pareto-optimal solutions for constrained optimization. The results obtained for both unconstrained and constrained optimization amply demonstrate that the EM-MOPSO technique can yield efficient Pareto-optimal solutions for multi-objective optimization.

6. Discussion

The proposed approach is simple to implement, yet efficient in yielding true Pareto-optimal solutions. The computational complexity is also reasonable. In this approach, in addition to the objective function computations, the computational complexity of the algorithm is mainly governed by the non-dominated comparison of the particles in the population, sorting, and crowding distance computation. If there are m objective functions and N solutions (particles) in the population, the objective function computation has $O(mN)$ computational complexity. The costly part of crowding distance computation is sorting the solutions in each objective function. Sorting K solutions in the external repository has $O(mK \log K)$ computational complexity. If the population and the external repository have the same number of solutions, say N, the computational complexity for the non-dominated comparison is $O(mN^2)$. Thus the overall complexity of the proposed EM-MOPSO is $O(mN^2)$.

The parameters of interest and their sensitivity are briefly discussed below.

- *Population size and maximum number of iterations*. In general these two parameters have an inverse relation, as a higher population size generally requires fewer iterations and vice versa. In this study, a population size of 100 is used for all the test problems. However, depending on the complexity of the problem, the maximum number of iterations is varied to run the algorithm. A large population size may be required to achieve a good performance of the algorithm for more complex problems.
- *Size of the external repository (ERP)*. The algorithm allows flexibility in setting the size of the external repository to any maximum value. As the *ERP* size increases, the computing requirement becomes enormous (as it is required for sorting and crowding value calculations). Hence, in order to reduce additional computational cost in iteration, this should be restricted to a desirable value, without affecting the performance of the algorithm. In this study, for effectiveness, a variable-size external repository was chosen. The size was initially set to 10% of maximum *ERP*, and then increased in a stepwise manner until it reached the maximum *ERP* at the start of 90% of maximum iterations. This strategy worked quite well for all the test problems considered in this study. Therefore this approach is recommended to save computational time and to improve the performance of the algorithm.
- *Size of elitist-mutated particles*. To experiment with the size of elitist-mutated particles, tests were conducted with 5, 10, 15, and 20 particles for problems with a maximum population

of 100. It should be noted that this also depends on the size of the population. The number of particles to be elitist-mutated is preferably $\leq$20% of the population size to ensure that the population should not lose control at the cost of exploring better non-dominated solutions. The probability of elitist-mutated particles is fixed at a certain value after a few trials. However, the mutation scale used in this study was varied linearly from 0.2 to 0.01 over the iterations. This was selected after a few trials to ensure that the search would not stagnate at an initial stage, and that there should be no deterioration while exploring the search space. The strategy adopted worked quite well for all the test problems and can be used for similar complex problems.

7. Conclusions

An efficient procedure for solving multi-objective optimization problems using swarm intelligence principles has been presented. The proposed algorithm for multi-objective particleswarm optimization (MOPSO) combines Pareto dominance criteria for non-domination selection, an external repository for elitism, and a crowding distance comparison operator for promoting solution diversity. The performance of the algorithm is enhanced by effectively exploring the search space by incorporating an efficient mutation strategy called elitist-mutation. This strategy helps the exploration and exploitation of the search space for feasible non-dominated solutions. The proposed approach was tested on various benchmark problems, having complexity in the search space with convex and non-convex objective functions. The algorithm was also applied to three engineering design problems to demonstrate its applicability in practical problems. The results obtained amply demonstrate that the approach is efficient in converging to the true Pareto fronts and finding a diverse set of solutions along the Pareto front. Thus the proposed EM-MOPSO algorithm can be effectively applied to real-world multi-objective optimization problems to arrive at efficient solutions.

References

Coello, C.A.C. and Lechuga, M.S., MOPSO: A proposal for multiple objective particle swarm optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC'2002)*, Honolulu, HI, 2002, Vol. 1, 1051–1056.

Coello C.A.C., Pulido, G.T. and Lechuga, M.S., Handling multiple objectives with particle swarm optimization. *IEEE Trans. Evol. Comput.*, 2004, **8**, 256–279.

Deb, K., *Multi-objective Optimization Using Evolutionary Algorithms*, 2001 (John Wiley: Chichester).

Deb, K., Pratap, A. and Moitra, S., Mechanical component design for multiple objectives using elitist non-dominated sorting GA. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*,Paris, 16–20 September 2000, pp. 859–868.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T., A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, 2002, **6**, 182–197.

Fieldsend, J.E. and Singh, S., A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence. In *Proceedings of the 2002 UK Workshop on Computational Intelligence*, Birmingham, 2002, pp. 37–44.

Fonseca, C.M. and Fleming, P.J., Multiobjective optimization and multiple constraint handling with evolutionary algorithms. Part II: Application example. *IEEE Trans. Systems Man Cybernet.*, 1998, **28**, 38–47.

Hu, X. and Eberhart, R., Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC'2002)*, Honolulu, HI, 2002, Vol. 2, 1677–1681.

Hu, X., Eberhart, R.C. and Shi, Y., Particle swarm with extended memory for multiobjective optimization. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, Indianapolis, IN, 2003, pp. 193–197.

Kennedy, J. and Eberhart, R.C., *Swarm Intelligence*, 2001 (San Mateo, CA: Morgan Kaufmann).

Knowles, J.D. and Corne, D.W., Approximating the nondominated front using the Pareto archived evolution strategy, *Evol. Comput.*, 2000, **8**, 149–172.

Kumar, D.N. and Reddy, M.J., Multipurpose reservoir operation using particle swarm optimization. *J. Water Resour. Plan. Manage. ASCE*, 2006, in print.

Li, X., A nondominated sorting particle swarm optimizer for multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2003*, Part I, edited by E. Cantú-Paz *et al.*, 2003, pp. 37–48 (Springer: Berlin).

Palli, N., Azram, S., McCluskey, P. and Sundararajan, R., An interactive multistage ε-inequality constraint method for multiple objectives decision making. *ASME J. Mech. Des.*, 1999, **120**(4), 678–686.

Parsopoulos, K.E. and Vrahatis, M.N., Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'2002)*, Madrid, 2002, pp. 603–607.

Ray, T. and Liew, K.M., A swarm metaphor for multiobjective design optimization. *Eng. Optimiz.*, 2002, **34**(2), 141–153.

Schaffer, J. D., Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, edited by JJ. Grefensttete, 1987, pp. 93–100 (Hillsdale, NJ: Lawrence Erlbaum).

Veldhuizen, D.V., Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. PhD Thesis, Air Force Institute of Technology, Dayton, OH, 1999 (Tech. Rep. AFIT/DS/ENG/99-01).

Yang, B.S., Yeun, Y.-S. and Ruy, W.-S., Managing approximation models in multiobjective optimization, *Struct. Multidisc. Optimiz.*, 2002, **24**, 141–156.

Zitzler, E., Evolutionary algorithms for multiobjective optimization: methods and applications. Doctoral Dissertation, Swiss Federal Institute of Technology (ETH), Zurich, 1999.

Zitzler, E. and Thiele, L., Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.*, 1999, **3**, 257–271.

Zitzler, E., Deb, K. and Thiele, L., Comparison of multi-objective evolutionary algorithms: empirical results. *Evol. Comput.*, 2000, **8**(2), 125–148.