A Memetic Algorithm for Dynamic Multiobjective Optimization

Tapabrata Ray, Amitay Isaacs, and Warren Smith

University of New South Wales, Australian Defence Force Academy,
ACT 2600, Australia
{t.ray,a.isaacs,w.smith}@adfa.edu.au

Summary. Dynamic multi-objective optimization (DMO) is one of the most challenging class of multi-objective optimization problems where the objective function(s) change over time and the optimization algorithm is required to identify the corresponding Pareto optimal solutions with minimal time lag. DMO has received very little attention in the past and none of the existing multi-objective algorithms perform too well on the set of newly proposed DMO test problems. In this chapter, we introduce a memetic algorithm (MA) to deal with such class of problems. The memetic algorithm employs an orthogonal epsilon-constrained formulation to deal with multiple objectives and a sequential quadratic programming (SQP) solver is embedded as its local search mechanism to improve the rate of convergence. The performance of the memetic algorithm is compared with an evolutionary algorithm (EA) embedded with a Sub-EA on two benchmarks FDA1 and modified FDA2. The memetic algorithm consistently outperforms the evolutionary algorithm with Sub-EA for both FDA1 and modified FDA2 for all problem sizes. A variational study on the effect of the number of SQP iterations on the behavior of MA is included to highlight the benefits offered by MA for such class of problems.

Keywords: dynamic multi-objective optimization, memetic algorithm, evolutionary algorithm, orthogonal epsilon-constrained method.

1 Introduction

Over the years, a number of optimization algorithms such as evolutionary algorithms (EA), particle swarm optimization (PSO), differential evolution (DE) and hybrids such as memetic algorithms have been successfully used to solve a number of real life multi-objective optimization problems. However, most of the reported applications deal with static multi-objective optimization problems, where the objective functions do not change over time. For such classes of problems, the aim of an optimization algorithm is to identify a diverse set of non-dominated solutions that are close to the Pareto optimal front.

Dynamic multi-objective optimization is far more challenging as compared to its static counterpart. In order to solve a DMO problem effectively and efficiently, the optimization algorithm should posses mechanisms to (a) identify a change in the objective function(s), (b) converge to the Pareto optimal set with a high rate of convergence, and (c) allow migration from a converged Pareto optimal set to another.

In terms of identifying a change in the objective function, there are basically two approaches reported in literature:- proactive and reactive. In proactive approach, a forecasting model is developed based on objective function behavior over time and a portion of the population is evolved based on an objective function derived through the forecasting model (1). Such an approach is suitable if the objective function follows a pattern over time as in many DMO test problems but may not be the best choice for all classes of problems. The reactive approach on the other hand responds to a change in the objective function only when it is detected and is a more generic approach to solving DMO problems. Performance of reactive model on some test cases under unpredictable parameter variations is reported in (2).

In order to solve a DMO problem efficiently, the underlying optimization algorithm should have a high rate of convergence. A hybrid search i.e. a combination of a global and a local search is particularly attractive for such problems as the rate of convergence of a hybrid search is better than a global search alone. Memetic algorithm (MA) (3; 4) is one such hybrid which is known for its high rate of convergence. MA integrates the global search properties of a population based approach with the exploitation properties of a local search algorithm. An excellent review of memetic algorithms has been presented by (5). However, the performance of MA is largely dependent on the correct choice of the local search strategies (memes), identification of the subset undergoing local improvements and the convergence criterion used in local search strategies. Furthermore, for MO problems in general, the MA should posses appropriate mechanisms to maintain diversity and an acceptable level of spread along the Pareto optimal front. Epsilon constraint formulation (6) is an attractive choice to deal with the above problem but needs a prudent choice of the constraints to uncover regions of the non-convex front. Adaptive GA with dynamic fitness function guided by fuzzy inference system to control crossover and mutation rates has been reported in (7).

The third aspect of DMO algorithm is particularly difficult to incorporate as migration from a converged Pareto optimal set to another may require substantial traversal in the variable space and thus lead to a larger time lag. A generic approach would be to use a random population instead of the converged Pareto solutions whenever a change in the objective function is detected. However, such an approach is likely to have larger time lag as compared to a migration from the converged Pareto set for problems where there is gradual migration of the variables. The recombination and mutation schemes employed within an EA are also expected to influence the convergence behavior.

In this chapter, we present a memetic algorithm that consists of an EA coupled with a SQP solver to deal with DMO problems. An orthogonal epsilon-constrained formulation is used to deal with multiple objectives where orthogonal sets of constraints are used to uncover all parts of the Pareto optimal front. The underlying EA is embedded with a SQP solver for faster convergence. The MA with orthogonal epsilon constraint formulation attempts to solve a series of single objective constrained optimization problems to yield the set of non-dominated solutions. The details of the algorithm are outlined in Section 2 while its performance on the set of mathematical benchmarks FDA1 and modified FDA2 are presented in Section 3 followed by a summary of our findings in Section 4.

2 Memetic Algorithm for Dynamic MO Optimization

The framework for the proposed memetic algorithm is outlined in Algorithm 1. The structure is identical to an evolutionary algorithm framework with the only exception that MA utilizes gradient evolve mechanism while an EA uses Sub-EA evolve mechanism.

Algorithm 1. Memetic / Evolutionary algorithm for DMO problems

```
Require: $N_G > 1$                                  /* Number of Generations */
Require: $N > 0$                                    /* Population size */
 1: $P_1$ = Initialize()
 2: Evaluate($P_1$)
 3: for $i = 2$ to $N_G$ do
 4:    if the function has changed then
 5:       Evaluate($P_{i-1}$)
 6:    end if
 7:    $C_{i-1}$ = GradientEvolve($P_{i-1}$) or $C_{i-1}$ = SubEAEvolve($P_{i-1}$)
 8:    Evaluate($C_{i-1}$)
 9:    $P_i$ = Reduce($P_{i-1} + C_{i-1}$)
10: end for
```

2.1 Initialization

The solutions in the initial population are initialized by selecting individual variable values in the specified range as given in Eq. 1.

$$x_i = \underline{x_i} + \mathcal{U}[0,1]\ (\overline{x_i} - \underline{x_i}) \qquad 1 \leq i \leq n \tag{1}$$

where x_i denotes the initialized variable, $\underline{x_i}$ and $\overline{x_i}$ are the lower and upper limits for ith variable respectively, n is the size of the design variable vector and $\mathcal{U}[0,1]$ is a uniform random number lying between 0 and 1.

2.2 Evaluation

The solutions in a population are evaluated using disciplinary analysis to calculate the objectives and the constraints. For dynamic problems, the function (objectives and/or constraints) behavior changes with time. The objective and the constraint values of a candidate solution calculated in the previous generation may not be valid for the next generation. To check if the objective and/or constraint functions have changed with time, a solution is picked randomly from the population and re-evaluated. If the objective or the constraint values have changed, then the function behavior is assumed to have changed and the entire population is re-evaluated to obtain new values of the objectives and the constraints.

2.3 Gradient Evolve Method

The gradient evolve mechanism employs a sequential quadratic programming (SQP) solver to solve a series of single objective optimization problems generated through an orthogonal epsilon-constrained formulation of the multi-objective problem. Consider a bi-objective optimization problem as presented in Eq. 2.

$$\text{Minimize } f_1(\mathbf{x}),\ f_2(\mathbf{x}) \quad \text{subject to} \quad g_i(\mathbf{x}) \le 0, \quad i = 1, \ldots, m \tag{2}$$

The extent of the Pareto optimal front is established by 2 separate single objective optimization problems given in Eq. 3.

$$\begin{aligned} &\text{Minimize } f_1(\mathbf{x}) \quad \text{subject to} \quad g_i(\mathbf{x}) \le 0, \quad i = 1, \ldots, m \\ &\text{Minimize } f_2(\mathbf{x}) \quad \text{subject to} \quad g_i(\mathbf{x}) \le 0, \\ &\quad i = 1, \ldots, m \end{aligned} \tag{3}$$

The solutions of the single optimization problems ($f_1^{\min}$ and $f_2^{\min}$) correspond to the extremities of the Pareto optimal front as shown in Figure 1. The rest of the Pareto optimal front is obtained by orthogonal epsilon-constrained reformation for each objective. The range of each objective is sub-divided into small intervals and a single objective optimization problem is solved for each of those

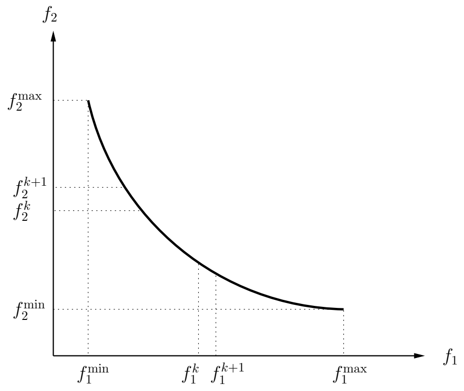


Fig. 1. Pareto optimal front for bi-objective optimization problem and formulation of orthogonal epsilon-constrained formulation

intervals. For $f_1(\mathbf{x})$, the range of values is $[f_1^{\min}, f_1^{\max}]$ and is sub-divided into equal intervals $[f_1^k, f_1^{k+1}]$ as shown in Figure 1. For each interval $[f_1^k, f_1^{k+1}]$, a single objective optimization problem is solved as given in Eq. 4.

$$\begin{aligned} \text{Minimize} \quad & f_2(\mathbf{x}) \\ \text{subject to} \quad & f_1^k \le f_1(\mathbf{x}) \le f_1^{k+1} \\ & g_i(\mathbf{x}) \le 0, \quad i = 1, \ldots, m \end{aligned} \tag{4}$$

Similarly, the range for $f_2(\mathbf{x})$ is $[f_2^{\min}, f_2^{\max}]$ and the single optimization problem is solved on each interval $[f_2^k, f_2^{k+1}]$ (Eq. 5).

$$\begin{aligned} \text{Minimize} \quad & f_1(\mathbf{x}) \\ \text{subject to} \quad & f_2^k \le f_2(\mathbf{x}) \le f_2^{k+1} \\ & g_i(\mathbf{x}) \le 0, \quad i = 1, \ldots, m \end{aligned} \tag{5}$$

Both, the convex and the non-convex parts of the Pareto optimal front can be captured using such an approach.

Solution of a single objective optimization problem requires a starting point. For each of the single optimization problems, a solution is picked up from the parent population randomly as the starting point. The number of function evaluations are dictated by the number of SQP iterations.

2.4 Sub-EA Evolve Method

In the Sub-EA evolve mechanism, the parent population of EA is allowed to evolve using an embedded evolutionary algorithm (Sub-EA) with conventional crossover and mutation operators. The main steps of the Sub-EA evolve method are given in Algorithm 2. In the present study, simulated binary crossover (SBX) and polynomial mutation operators are used (8). The population size for sub-EA is the same as that of the EA. For each generation of EA, multiple generations are evolved using sub-EA. The number of function evaluations is dictated by the number of generations of sub-EA.

Algorithm 2. Sub-EA Evolution Algorithm

```
Require: $N'_G > 1$                    /* Number of Sub-EA Generations */
Require: $P_j$                         /* Parent Population of jth generation */
1: $P'_1 = P_i$
2: Evaluate($P'_1$)
3: for $i = 2$ to $N'_G$ do
4:     $C'_{i-1}$ = Crossover($P'_{i-1}$)
5:     $C'_{i-1}$ = Mutation($C'_{i-1}$)
6:     Evaluate($C'_{i-1}$)
7:     $P'_i$ = Reduce($P'_{i-1} + C'_{i-1}$)
8: end for
9: $C_j = P'_{N'_G}$       /* Sub-EA evolved population is offspring population for EA */
```

2.5 Reduction

The reduction process retains N elite solutions for the next generation from a set of $2N$ solutions (parent and offspring population). Non-dominated sorting is used to sort $2N$ solutions into non-dominated fronts and within each front, the solutions are ranked using crowding distance sort (9). The reduction procedure is as follows.

1. If there are N or more feasible solutions,
   - N feasible solutions are selected in the order of non-dominated fronts and decreasing order of crowding distance in each front.
2. If there are less than N feasible solutions,
   - all the feasible solutions are selected, and
   - remaining solutions are selected from infeasible solutions in the increasing order of maximum constraint violation value.

Non-dominated sorting and crowding distance based sorting helps maintain the diversity in the population and spreads the solutions along the Pareto front.

3 DMO Test Problems

A static multi-objective optimization problem has fixed Pareto Optimal Front (POF) and corresponding solutions in the variable space denoted by POS. For dynamic multi-objective optimization problems POF and/or POS vary with time. A classification of multi-objective optimization problems are presented by Farina *et. al.* (10). Test problems FDA1 (10) and modified FDA2 (11) are used for this study. The FDA1 test problem is as defined by Eq. 6.

$$\text{FDA1:} \quad \begin{cases} f_1(\mathbf{x}_I) = x_1, \\ f_2(\mathbf{x}_{II}) = g \times h, \\ g(\mathbf{x}_{II}) = 1 + \sum_{x_i \in \mathbf{x}_{II}} (x_i - G(t))^2 \\ h(f_1, g) = 1 - \sqrt{f_1/g} \\ G(t) = \sin(0.5\pi t), \quad t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_T} \right\rfloor \\ \mathbf{x}_I = (x_1) \in [0,1], \\ \mathbf{x}_{II} = (x_2, \ldots, x_n) \in [-1,1]. \end{cases} \tag{6}$$

where τ is the generation counter, τ_T is the number of generations for which t remains fixed, and n_t is the number of distinct steps in t. The values used for the variables are: $n = 11$, $n_t = 10$, and $\tau_T = 5$.

Every time t changes, the POS changes but the POF remains same corresponding to $f_2 = 1 - \sqrt{f1}$. At the first time instant $t = 0$, the POS solutions correspond to $x_i = 0$ for $x_i \in \mathbf{x}_{II}$. With changes in t, each variable in $\mathbf{x}_{II}$ changes in a sinusoidal manner corresponding to a change in $G(t)$ as seen in Figure 2.

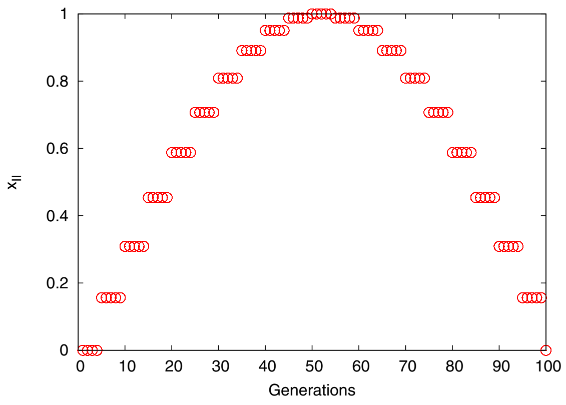


Fig. 2. The variation of variable values in x_{II} with time for FDA1

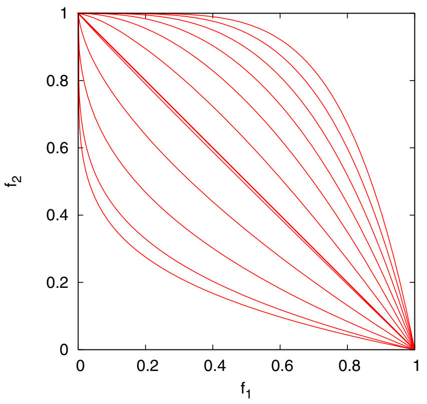


Fig. 3. FDA2$_{mod}$ POF changes from a convex shape to a concave shape with time

The definition for modified FDA2 (FDA2$_{mod}$) is given in Eq. 7.

$$\text{FDA2}_{mod}: \begin{cases} f_1(\mathbf{x}_I) = x_1, \\ f_2(\mathbf{x}_{II}, \mathbf{x}_{III}) = g \times h \\ g(\mathbf{x}_{II}, \mathbf{x}_{III}) = 1 + \sum_{x_i \in \mathbf{x}_{II}} x_i^2 + \sum_{x_i \in \mathbf{x}_{III}} (x_i + 1)^2 \\ h(f_1, g) = 1 - \left(\frac{f_1}{g}\right)^{H(t)} \\ H(t) = 0.2 + 4.8t^2, \quad t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_T} \right\rfloor \\ \mathbf{x}_I = (x_1) \in [0, 1], \\ \mathbf{x}_{II} = (x_2, \ldots, x_{r_1}) \in [-1, 1], \\ \mathbf{x}_{III} = (x_{r_1+1}, \ldots, x_n) \in [-1, 1]. \end{cases} \tag{7}$$

where τ is the generation counter, τ_T is the number of generations for which t remains fixed, and n_t is the number of distinct steps in t. The values used for the variables are: $n = 31$, $r_1 = 16$, $n_t = 10$, and $\tau_T = 5$. The POF for FDA2$_{mod}$ changes from a convex shape to a concave shape as t varies as shown in Figure 3.

A series of performance metrics for multi-objective optimization have been reported in literature (12; 13). These performance metrics use various measures on the non-dominated set of solutions such as cardinality, distance, volume, spread, *etc.* For the test problems used in this study, the POF and the POS variations are known and performance metrics based on absolute measures can be used. One such metric is the generational distance. The generational distance is the measure of the distance between the Pareto optimal front and the non-dominated solution set (14). The metric is defined as follows:

Let F^* denote the Pareto Optimal Front (POF) and S^* denote the Pareto Optimal Set (POS). The generational distance of a set of non-dominated solutions (F) with respect to the POF $F*$ is given by,

$$G(F, F^*) = \frac{1}{|F|} \left(\sum_{i=1}^{n} (d_i)^p \right)^{\frac{1}{p}},$$

where d_i is the Euclidean distance between the ith non-dominated solution of F and the nearest member of the Pareto front F^*, and $|F|$ is the number of elements in the non-dominated set. Most often $p = 2$. The same metric can also be used in the variable space to measure the generational distance of the non-dominated solutions in the variable space (S) with respect to the POS S^*.

$$G(S, S^*) = \frac{1}{|S|} \left(\sum_{i=1}^{n} (d_i)^p \right)^{\frac{1}{p}},$$

4 Results

4.1 Experimental Setup

A population size of 40 is evolved over 100 generations for both MA and EA. Each generation corresponds to a single time step. Both the algorithms are run for 100 time steps. For EA, the parameters of sub-EA (crossover and mutation parameters) are varied along with the random seed, resulting in a total of 32

Table 1. Parameters Values for 32 experimental runs of EA

Random Seed	10, 20
Crossover Probability	0.8, 0.9
Crossover Distribution Index	10, 20
Mutation Probability	0.05, 0.1
Mutation Distribution Index	20, 50

experimental runs. Since there are no additional parameters for MA, 32 different values of random seed are chosen. Shown in Table 1 are the parameter values used for various runs for EA. The random seed values for MA used are $10, 20, \ldots, 320$.

To ensure that EA and MA use the same number of function evaluations, first an estimate of number of function evaluations used by MA is obtained based on few runs. Gradient based algorithm uses a tolerance criterion to terminate the algorithm, or alternately, it can use a fixed iteration count. For MA the number of SQP iterations is fixed at 2 and the corresponding number of function evaluations is used to determine the number of SubEA generations. For FDA1, function evaluations corresponding to 2 iterations of SQP are equivalent to 70 generations of sub-EA. For FDA2$_{mod}$, function evaluations corresponding to 2 iterations of SQP are equivalent to 190 generations of SubEA.

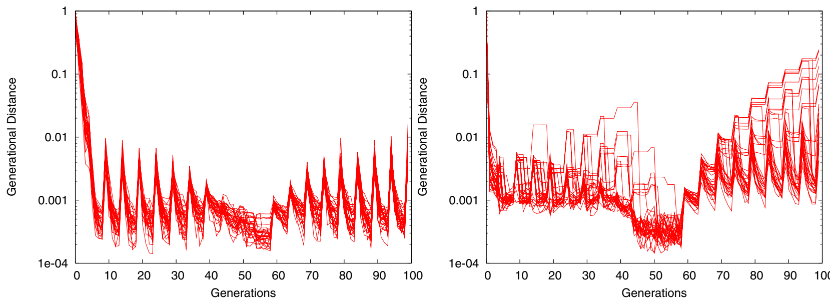
4.2 Results of FDA1

The statistics of multiple runs for MA and EA are presented in Table 2. EA uses a fixed number of function evaluations as dictated by the population size and the number of generations. As MA uses gradient based search, the number of function evaluations vary across multiple runs. The difference in the minimum and maximum number of function evaluations for MA is 384 (283,167-282,783) which is approximately 0.1% of the average number of function evaluations. Thus the number of function evaluations for MA and EA are considered equivalent.

The performance of MA and EA is compared using the generational distance metric. Variation in the generational distance for POF across multiple runs of

Table 2. Function evaluations statistics for MA and EA for FDA1

	Min.	Avg.	Max.
MA	282,783	283,001	283,167
EA	282,099	282,099	282,099



(a) MA (b) EA

Fig. 4. FDA1: Generational Distance metric for the Pareto optimal fronts (POF) obtained by MA and EA for all the experimental runs

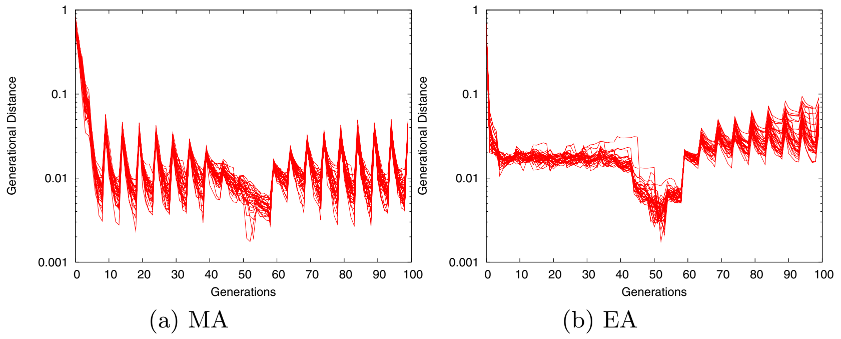


Fig. 5. FDA1: Generational Distance metric for the Pareto optimal sets(POS) obtained by MA and EA for all the experimental runs

MA and EA are shown in Figure 4. The POF generational distance is quite high in the beginning and reduces in next few generations. At every interval of $t = 5$, there is a sudden increase in the generational distance as seen in Figure 4(a). This increase corresponds to the change in the function form for $t = 5$. The POF generational distance using MA is consistently lower than EA across the generations. Since the movement of POS is very small between time steps 40 and 60, the POF generational distance for EA and MA shows a dip as in Figure 4(b).

Shown in Figure 5 is the variation in the POS generational distance across multiple runs of MA and EA. The POS generational distance in the variable space is calculated for the solutions that are non-dominated in objective space. The POS generational distance using MA (Figure 5(a)) shows a similar trend as that of POF generational distance (Figure 4(a)). Small movement of POS during the time steps 40 and 60 is visible prominently in the POS generational distance using EA as shown in Figure 5(b).

When the SQP search in MA is allowed to run for more number of iterations, the solutions converge closer to the Pareto front as seen in Figure 6(a) and

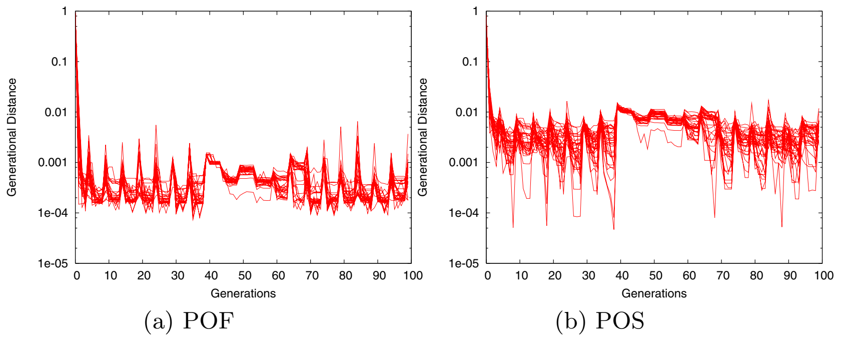


Fig. 6. FDA1: Generational Distance metric for the POF and the POS obtained by MA when the SQP search is run for 10 iterations

Table 3. Function evaluations used by MA with SQP search executed for 2 and 10 iterations for FDA1

SQP Iterations	Function evaluations		
	Min.	Avg.	Max.
2	282,783	283,001	283,167
10	492,500	497,568	501,678

Table 4. Function Evaluations used by MA and EA for FDA1 with different number of design variables

	MA			EA	
dim.	Min	Avg.	Max	Avg.	SubEA gens.
11	282,849	282,996	283,091	282,099	70
15	375,434	375,561	375,644	361,299	90
20	491,083	491,271	491,446	480,099	120
25	606,786	606,995	607,240	598,899	150

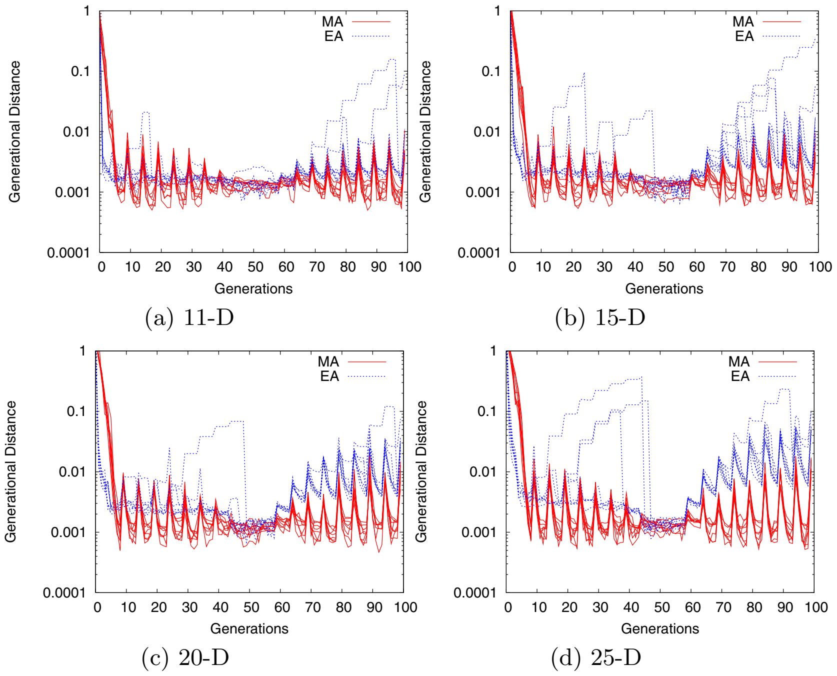


Fig. 7. FDA1: Generational Distance metric for the POF obtained by MA and EA for different number of design variables

(Figure 6(b)). The improvement in performance of MA is at the cost of increased number of function evaluations as seen from Table 3.

4.3 Results of FDA1 Problem Size Variation

Increasing the problem size has an immediate effect on the number of function evaluations used by MA as the gradient calculation requires additional evaluations. The number of function evaluations used by MA for different number of design variables (change in the dimension of x_{II}) is given in Table 4. The equivalent number of function evaluations for EA are obtained by increasing the number of sub-EA generations as given in Table 4.

The effects of increase in the number of design variables on the POF generational distance can be observed through a comparison between Figure 4 with Figure 7 and on the POS generational distance can be seen through a comparison between Figure 5 with Figure 8. MA consistently outperforms EA for all the problem sizes in both POF and POS.

4.4 Results of FDA2

The POF generational distance variation by MA and EA for FDA2$_{mod}$ is presented in Figure 9. Its clear that MA is able to maintain lower generational

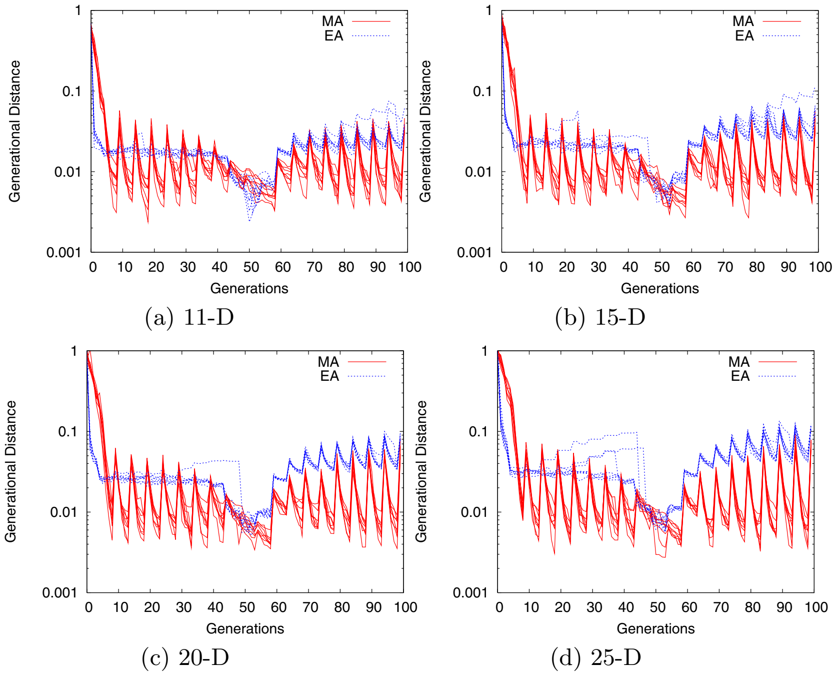
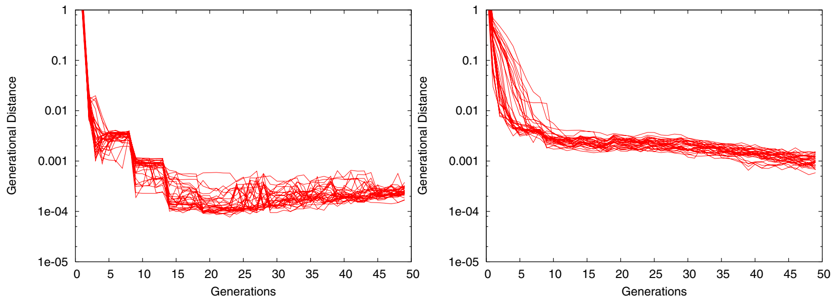
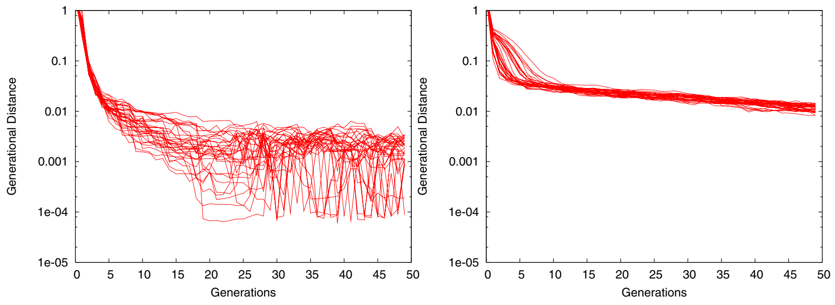


Fig. 8. FDA1: Generational Distance metric for the POS obtained by MA and EA for different number of design variables



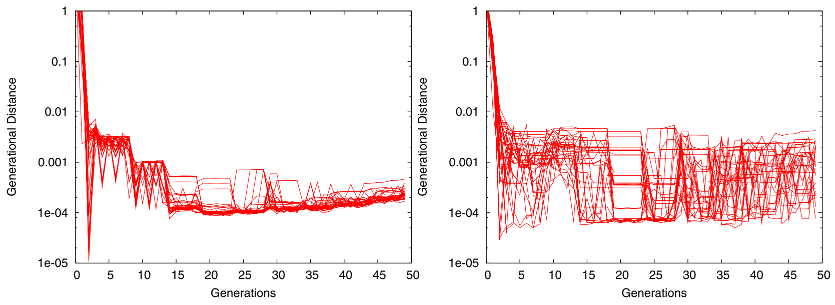
(a) MA (b) EA

Fig. 9. FDA2$_{mod}$: Generational Distance metric for the Pareto optimal fronts (POF) obtained by MA and EA for all the experimental runs



(a) MA (b) EA

Fig. 10. FDA2$_{mod}$: Generational Distance metric for the Pareto optimal sets (POS) obtained by MA and EA for all the experimental runs



(a) MA (b) EA

Fig. 11. FDA2$_{mod}$: Generational Distance metric for the POF and the POS obtained by MA when the SQP search is run for 10 iterations

distance as compared to EA over the generations. The corresponding POS generational distance for MA and EA is presented in Figure 5. It is interesting to observe that although POS generational distance for MA is lower than EA, it exhibits a significant fluctuation whereas EA presents a relatively flat behavior.

The generational distance variation (POF and POS) for MA with an increase in the number of SQP iterations from 2 to 10 is presented in Figure 11. An increase in the number of SQP iterations, improves the generational distance in both POF and POS as observed for FDA1.

5 Summary and Conclusion

In this chapter, a memetic algorithm is presented for dynamic multi-objective optimization. Currently, DMO problems are considered intractable and there is considerable interest to develop optimization methods to solve such classes of problems. A memetic algorithm is proposed to solve DMO problems and the results on the test functions clearly indicate its superiority over EA based approaches. The memetic algorithm is embedded with a sequential quadratic programming (SQP) solver as a local search mechanism for an improved rate of convergence. An orthogonal epsilon-constrained formulation is used to reformulate a multi-objective optimization problem as series of single objective optimization problems and uncover both convex and non-convex part of the Pareto optimal front. The algorithm is designed as a reactive model, where it acts on migration only when a change is detected in the objective function. Upon detecting a change in the objective function, existing solutions are used as starting points for SQP search although a random start points for SQP is also a possibility. The performance of the proposed MA is compared with an evolutionary algorithm undergoing secondary evolution using Sub-EA for two standard benchmarks FDA1 and modified FDA2 over a range of problem dimensions and the number of SQP iterations. The results clearly indicate that MA outperforms EA for all problem sizes for the same computational cost delivering solutions that are closer to the Pareto optimal front in both the objective and variable space and stands out as a promising alternative for DMO problems. The algorithm is currently being tested for online UAV system identification and control.

References

[1] Hatzakis, I., Wallace, D.: Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach, Seattle, Washington, USA, pp. 1201–1208. ACM, New York (2006)

[2] Amato, P., Farina, M.: An ALife-inspired evolutionary algorithm for dynamic multiobjective optimization problems (2003)

[3] Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical report, Caltech (1989)

[4] Ong, Y.S., Keane, A.: Meta-lamarckian learning in memetic algorithms. IEEE Transactions on Evolutionary Computation 8, 99–110 (2004)

[5] Krasnogor, N., Smith, J.: Multimeme algorithms for the structure prediction and structure comparison of proteins. In: GECCO, pp. 42–44 (2002)
[6] Chankong, V., Haimes, Y.Y.: Multiobjective Decision Making: Theory and Methodology. North-Holland, Amsterdam (1983)
[7] Bingul, Z., Sekmen, A., Zein-Sabatto, S.: Adaptive genetic algorithms applied to dynamic multiobjective problems. In: Artifiticial Neural Networks Engineering Conference, pp. 273–278 (2000)
[8] Deb, K., Agrawal, S.: Simulated binary crossover for continuous search space. Complex Systems 9, 115–148 (1995)
[9] Deb, K., Agrawal, 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, pp. 849–858 (2000)
[10] Farina, M., Deb, K., Amato, P.: Dynamic multiobjective optimization problems: test cases, approximations, and applications. IEEE Transactions on Evolutionary Computation 8, 425–442 (2004)
[11] Mehnen, J., Wagner, T., Rudolph, G.: Evolutionary optimization of dynamic multi-objective test functions. In: Proceedings of the Second Italian Workshop on Evolutionary Computation (GSICE2), Siena, Italy (September 2006)
[12] Veldhuizen, D.V., Lamont, G.: On measuring multiobjective evolutionary algorithm performance. In: Proceedings of the Congress on Evolutionary Computation (CEC) 2000, vol. 1, pp. 204–211 (2000)
[13] Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8, 173–195 (2000)
[14] Van Veldhuizen, D.A., Lamont, G.B.: Multiobjective evolutionary algorithm test suites. In: Proceedings of the 1999 ACM Symposium on Applied Computing, pp. 351–357 (1999)