A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II

Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T Meyarivan

Kanpur Genetic Algorithms Laboratory (KanGAL)
Indian Institute of Technology Kanpur
Kanpur, PIN 208 016, India
{deb,samira,apratap,mary}@iitk.ac.in
http://www.iitk.ac.in/kangal

KanGAL Report No. 200001

Abstract. Multi-objective evolutionary algorithms which use non-dominated sorting and sharing have been mainly criticized for their (i) $O(mN^3)$ computational complexity (where m is the number of objectives and N is the population size), (ii) non-elitism approach, and (iii) the need for specifying a sharing parameter. In this paper, we suggest a non-dominated sorting based multi-objective evolutionary algorithm (we called it the Non-dominated Sorting GA-II or NSGA-II) which alleviates all the above three difficulties. Specifically, a fast non-dominated sorting approach with $O(mN^2)$ computational complexity is presented. Second, a selection operator is presented which creates a mating pool by combining the parent and child populations and selecting the best (with respect to fitness and spread) N solutions. Simulation results on five difficult test problems show that the proposed NSGA-II is able to find much better spread of solutions in all problems compared to PAES—another elitist multi-objective EA which pays special attention towards creating a diverse Pareto-optimal front. Because of NSGA-II's low computational requirements, elitist approach, and parameter-less sharing approach, NSGA-II should find increasing applications in the years to come.

1 Introduction

Over the past decade, a number of multi-objective evolutionary algorithms (MOEAs) have been suggested [9, 3, 5, 13]. The primary reason for this is their ability to find multiple Pareto-optimal solutions in one single run. Since the principal reason why a problem has a multi-objective formulation is because it is not possible to have a single solution which simultaneously optimizes all objectives, an algorithm that gives a large number of alternative solutions lying on or near the Pareto-optimal front is of great practical value.

The Non-dominated Sorting Genetic Algorithm (NSGA) proposed in Srinivas and Deb [9] was one of the first such evolutionary algorithms. Over the years, the main criticism of the NSGA approach have been as follows:

High computational complexity of non-dominated sorting: The non-dominated sorting algorithm in use uptil now is $O(mN^3)$ which in case of large population sizes is very expensive, especially since the population needs to be sorted in every generation.

Lack of elitism: Recent results [12, 8] show clearly that elitism can speed up the performance of the GA significantly, also it helps to prevent the loss of good solutions once they have been found.

Need for specifying the sharing parameter σ_{share}: Traditional mechanisms of insuring diversity in a population so as to get a wide variety of equivalent solutions have relied heavily on the concept of sharing. The main problem with sharing is that it requires the specification of a sharing parameter (σ_{share}). Though there has been some work on dynamic sizing of the sharing parameter [4], a parameterless diversity preservation mechanism is desirable.

In this paper, we address all of these issues and propose a much improved version of NSGA which we call NSGA-II. From the simulation results on a number of difficult test problems, we find that NSGA-II has a better spread in its optimized solutions than PAES [6]—another elitist multi-objective evolutionary algorithm. These results encourage the application of NSGA-II to more complex and real-world multi-objective optimization problems.

2 Elitist Multi-Objective Evolutionary Algorithms

In the study of Zitzler, Deb, and Theile [12], it was clearly shown that elitism helps in achieving better convergence in MOEAs. Among the existing elitist MOEAs, Zitzler and Thiele's [13] strength Pareto EA (SPEA), Knowles and Corne's Pareto-archived evolution strategy (PAES) [6], and Rudolph's [8] elitist GA are well known.

Zitzler and Thiele [13] suggested an elitist multi-criterion EA with the concept of non-domination in their strength Pareto EA (SPEA). They suggested maintaining an external population at every generation storing all non-dominated solutions discovered so far beginning from the initial population. This external population participates in genetic operations. At each generation, a combined population with the external and the current population is first constructed. All non-dominated solutions in the combined population are assigned a fitness based on the number of solutions they dominate and dominated solutions are assigned fitness worse than the worst fitness of any non-dominated solution. This assignment of fitness makes sure that the search is directed towards the non-dominated solutions. A deterministic clustering technique is used to ensure diversity among non-dominated solutions. Although the implementation suggested in [13] is $O(mN^3)$, with proper book-keeping the complexity of SPEA can be reduced to $O(mN^2)$. An important aspect of this study and subsequent studies [12, 11] is that they clearly show the importance of introducing elitism in evolutionary multi-criterion optimization.

Knowles and Corne [6] suggested a simple MOEA using an evolution strategy (ES). In their Pareto-archived ES (PAES) with one parent and one child, the child is compared with respect to the parent. If the child dominates the parent, the child is accepted as the next parent and the iteration continues. On the other hand, if the parent dominates the child, the child is discarded and a new mutated solution (a new child) is found. However, if the child and the parent do not dominate each other, the choice between the child and the parent considers the second objective of keeping diversity among obtained solutions. To maintain diversity, an archive of non-dominated solutions is maintained. The child is compared with the archive to check if it dominates any member of the archive. If yes, the child is accepted as the new parent and the dominated solution is eliminated from the archive. If the child does not dominate any member of the archive, both parent and child are checked for their *nearness* with the solutions of the archive. If the child resides in a least crowded region in the parameter space among the members of the archive, it is accepted as a parent and a copy of added to the archive. Later, they suggested a multi-parent PAES with similar principles as above. Authors have calculated the worst case complexity of PAES for N evaluations as $O(amN)$, where a is the archive length. Since the archive size is usually chosen proportional to the population size N, the overall complexity of the algorithm is $O(mN^2)$.

Rudolph [8] suggested, but did not simulate, a simple elitist multi-objective EA based on a systematic comparison of individuals from parent and offspring populations. The non-dominated solutions of the offspring population are compared with that of parent solutions to form an overall non-dominated set of solutions, which becomes the parent population of the next iteration. If the size of this set is not greater than the desired population size, other individuals from the offspring population are included. With this strategy, he has been able to prove the convergence of this algorithm to the Pareto-optimal front. Although this is an important achievement in its own right, the algorithm lacks motivation for the second task of maintaining diversity of Pareto-optimal solutions. An explicit diversity preserving mechanism must be added to make it more usable in practice. Since the determinism of the first non-dominated front is $O(mN^2)$, the overall complexity of Rudolph's algorithm is also $O(mN^2)$.

3 Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II)

The non-dominated sorting GA (NSGA) proposed by Srinivas and Deb in 1994 has been applied to various problems [10, 7]. However as mentioned earlier there have been a number of criticisms of the NSGA. In this section, we modify the NSGA approach in order to alleviate all the above difficulties. We begin by presenting a number of different modules that form part of NSGA-II.

3.1 A fast non-dominated sorting approach

In order to sort a population of size N according to the level of non-domination, each solution must be compared with every other solution in the population to find if it is dominated. This requires $O(mN)$ comparisons for each solution, where m is the number of objectives. When this process is continued to find the members of the first non-dominated class for all population members, the total complexity is $O(mN^2)$. At this stage, all individuals in the first non-dominated front are found. In order to find the individuals in the next front, the solutions of the first front are temporarily discounted and the above procedure is repeated. In the worst case, the task of finding of the second front also requires $O(mN^2)$ computations. The procedure is repeated to find the subsequent fronts. As can be seen the worst case (when there exists only one solution in each front) complexity of this algorithm is $O(mN^3)$. In the following we describe a fast non-dominated sorting approach which will require at most $O(mN^2)$ computations.

First, for each solution we calculate two entities: (i) n_i, the number of solutions which dominate the solution i, and (ii) S_i, a set of solutions which the solution i dominates. The calculation of these two entities requires $O(mN^2)$ comparisons. We identify all those points which have $n_i = 0$ and put them in a list $\mathcal{F}_1$. We call $\mathcal{F}_1$ the current front. Now, for each solution in the current front we visit each member (j) in its set S_i and reduce its n_j count by one. In doing so, if for any member j the count becomes zero, we put it in a separate list $\mathcal{H}$. When all members of the current front have been checked, we declare the members in the list $\mathcal{F}_1$ as members of the first front. We then continue this process using the newly identified front $\mathcal{H}$ as our current front.

Each such iteration requires $O(N)$ computations. This process continues till all fronts are identified. Since at most there can be N fronts, the worst case complexity of this loop is $O(N^2)$. The overall complexity of the algorithm now is $O(mN^2) + O(N^2)$ or $O(mN^2)$.

It is worth mentioning here that although the computational burden has reduced from $O(mN^3)$ to $O(mN^2)$ by performing systematic book-keeping, the storage has increased from $O(N)$ to $O(N^2)$ in the worst case.

The fast non-dominated sorting procedure which when applied on a population P returns a list of the non-dominated fronts $\mathcal{F}$.

`fast-nondominated-sort`(P)

for each $p \in P$	
for each $q \in P$	
if $(p \prec q)$ then	if p dominates q then
$S_p = S_p \cup \{q\}$	include q in S_p
else if $(q \prec p)$ then	if p is dominated by q then
$n_p = n_p + 1$	increment n_p
if $n_p = 0$ then	if no solution dominates p then
$\mathcal{F}_1 = \mathcal{F}_1 \cup \{p\}$	p is a member of the first front
$i = 1$	
while $\mathcal{F}_i \neq \emptyset$	
$\mathcal{H} = \emptyset$	
for each $p \in \mathcal{F}_i$	for each member p in $\mathcal{F}_i$
for each $q \in S_p$	modify each member from the set S_p
$n_q = n_q - 1$	decrement n_q by one
if $n_q = 0$ then $\mathcal{H} = \mathcal{H} \cup \{q\}$	if n_q is zero, q is a member of a list $\mathcal{H}$
$i = i + 1$	
$\mathcal{F}_i = \mathcal{H}$	current front is formed with all members of $\mathcal{H}$

3.2 Density Estimation

To get an estimate of the density of solutions surrounding a particular point in the population we take the average distance of the two points on either side of this point along each of the objectives. This quantity $i_{distance}$ serves as an estimate of the size of the largest cuboid enclosing the point i without including any other point in the population (we call this the *crowding distance*). In Figure 1, the crowding distance of the i-th solution in its front (marked with solid circles) is the average side-length of the cuboid (shown with a dashed box). The following algorithm is used to calculate the crowding

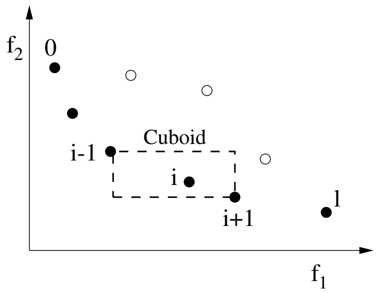

Fig. 1. The crowding distance calculation is shown.

distance of each point in the set $\mathcal{I}$:

`crowding-distance-assignment` $(\mathcal{I})$

$l = \|\mathcal{I}\|$	number of solutions in $\mathcal{I}$
for each i, set $\mathcal{I}[i]_{distance} = 0$	initialize distance
for each objective m	
$\mathcal{I}$ = sort$(\mathcal{I}, m)$	sort using each objective value
$\mathcal{I}[1]_{distance} = \mathcal{I}[l]_{distance} = \infty$	so that boundary points are always selected
for $i = 2$ to $(l-1)$	for all other points
$\mathcal{I}[i]_{distance} = \mathcal{I}[i]_{distance} + (\mathcal{I}[i+1].m - \mathcal{I}[i-1].m)$	

Here $\mathcal{I}[i].m$ refers to the m-th objective function value of the i-th individual in the set $\mathcal{I}$. The complexity of this procedure is governed by the sorting algorithm. In the worst case (when all solutions are in one front), the sorting requires $O(mN \log N)$ computations.

3.3 Crowded Comparison Operator

The crowded comparison operator ($\geq_n$) guides the selection process at the various stages of the algorithm towards a uniformly spread out Pareto-optimal front. Let us assume that every individual i in the population has two attributes.

1. Non-domination rank (i_{rank})
2. Local crowding distance ($i_{distance}$)

We now define a partial order $\geq_n$ as :

$i \geq_n j$ if $(i_{rank} < j_{rank})$ or $((i_{rank} = j_{rank})$ and $(i_{distance} > j_{distance}))$

That is, between two solutions with differing non-domination ranks we prefer the point with the lower rank. Otherwise, if both the points belong to the same front then we prefer the point which is located in a region with lesser number of points (the size of the cuboid inclosing it is larger).

3.4 The Main Loop

Initially, a random parent population P_0 is created. The population is sorted based on the non-domination. Each solution is assigned a fitness equal to its non-domination level (1 is the best level). Thus, minimization of fitness is assumed. Binary tournament selection, recombination, and mutation operators are used to create a child population Q_0 of size N. From the first generation onward, the procedure is different. The elitism procedure for $t \geq 1$ and for a particular generation is shown in the following:

$R_t = P_t \cup Q_t$	combine parent and children population
$\mathcal{F} =$ `fast-nondominated-sort` (R_t)	$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots)$, all non-dominated fronts of R_t
until $\|P_{t+1}\| < N$	till the parent population is filled
`crowding-distance-assignment` $(\mathcal{F}_i)$	calculate crowding distance in $\mathcal{F}_i$
$P_{t+1} = P_{t+1} \cup \mathcal{F}_i$	include i-th non-dominated front in the parent pop
Sort$(P_{t+1}, \geq_n)$	sort in descending order using $\geq_n$
$P_{t+1} = P_{t+1}[0:N]$	choose the first N elements of P_{t+1}
$Q_{t+1} =$ `make-new-pop` (P_{t+1})	use selection,crossover and mutation to create a new population Q_{t+1}
$t = t + 1$	

First, a combined population $R_t = P_t \cup Q_t$ is formed. The population R_t will be of size $2N$. Then, the population R_t is sorted according to non-domination. The new parent population P_{t+1} is formed by adding solutions from the first front till the size exceeds N. Thereafter, the solutions of the last accepted front are sorted according to $\geq_n$ and the first N points are picked. This is how we construct the population P_{t+1} of size N. This population of size N is now used for selection, crossover and mutation to create a new population Q_{t+1} of size N. It is important to note that we use a binary tournament selection operator but the selection criterion is now based on the niched comparison operator $\geq_n$.

Let us now look at the complexity of one iteration of the entire algorithm. The basic operations being performed and the worst case complexities associated with are as follows:

1. Non-dominated sort is $O(mN^2)$,
2. Crowding distance assignment is $O(mN \log N)$, and
3. Sort on $\geq_n$ is $O(2N \log(2N))$.

As can be seen, the overall complexity of the above algorithm is $O(mN^2)$.

The diversity among non-dominated solutions is introduced by using the crowding comparison procedure which is used in the tournament selection and during the population reduction phase. Since solutions compete with their crowding distance (a measure of density of solutions in the neighborhood), no extra niching parameter (such as σ_{share} needed in the NSGA) is required here. Although the crowding distance is calculated in the objective function space, it can also be implemented in the parameter space, if so desired [1].

It is interesting to note here the connection of this algorithm with the algorithm proposed by Rudolph [8]. Since the non-dominated front finding algorithm used in Rudolph's algorithm is $O(mN^2)$ for each front, Rudolph control's the complexity of his algorithm by working with just the first few fronts in the parent and the child populations and treating the rest of the individuals in the child population at par. With the availability of a fast non-domination sorting algorithm we can now afford to combine the parent and child populations and do a complete sort to identify all the fronts and allocate fitness accordingly.

4 Results

We compare NSGA-II with PAES on five test problems (minimization of both objectives):

$$\text{MOP2:} \begin{cases} f_1(\boldsymbol{x}) = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right) & -4 \le x_1, x_2, x_3 \le 4 \\ f_2(\boldsymbol{x}) = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right) & \end{cases} \tag{1}$$

$$\text{MOP3:} \begin{cases} f_1(\boldsymbol{x}) = \left[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2\right] \\ f_2(\boldsymbol{x}) = \left[(x+3)^2 + (y+1)^2\right] \end{cases} \tag{2}$$

$$\text{where} \quad \begin{aligned} A_1 &= 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2 \\ A_2 &= 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2 \\ B_1 &= 0.5\sin x - 2\cos x + \sin y - 1.5\cos y \\ B_2 &= 1.5\sin x - \cos x + 2\sin y - 0.5\cos y \end{aligned}$$

$$\text{MOP4:} \begin{cases} f_1(\boldsymbol{x}) = \sum_{i=1}^{n-1}\left(-10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right) & -5 \le x_1, x_2, x_3 \le 5 \\ f_2(\boldsymbol{x}) = \sum_{i=1}^{n}\left(|x_i|^{0.8} + 5\sin(x_i)^3\right) & \end{cases} \tag{3}$$

$$\text{EC4:} \begin{cases} f_1(\boldsymbol{x}) = x_1 & 0 \le x_1 \le 1 \\ f_2(\boldsymbol{x}) = g\left(1 - \sqrt{\frac{x_1}{g}}\right) & -5 \le x_2, \ldots, x_{10} \le 5 \end{cases} \tag{4}$$

$$\text{where} \quad g(\boldsymbol{x}) = 91 + \sum_{i=2}^{10}\left(x_i^2 - 10\cos(4\pi x_i)\right)$$

$$\text{EC6:} \begin{cases} f_1(\boldsymbol{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1) & 0 \le x_i \le 1 \quad i = 1, \ldots, 10 \\ f_2(\boldsymbol{x}) = g\left(1 - (f_1/g)^2\right) & \end{cases} \tag{5}$$

$$\text{where} \quad g(\boldsymbol{x}) = 1 + 9\left(\sum_{i=2}^{10} x_i/9\right)^{0.25}$$

Since the diversity among optimized solutions is an important matter in multi-objective optimization, we devise a measure based on the consecutive distances among the solutions of the best non-dominated front in the final population. The obtained set of the first non-dominated solutions are compared with a uniform distribution and the deviation is computed as follows:

$$\Delta = \sum_{i=1}^{|\mathcal{F}_1|} \frac{|d_i - \bar{d}|}{|\mathcal{F}_1|}. \tag{6}$$

In order to ensure that this calculation takes into account the spread of solutions in the entire region of the true front, we include the boundary solutions in the non-dominated front $\mathcal{F}_1$. For discrete Pareto-optimal fronts, we calculate a weighted average of the above metric for each of the discrete regions. In the above equation, d_i is the Euclidean distance between two consecutive solutions in the first non-dominated front of the final population in the objective function space. The parameter $\bar{d}$ is the average of these distances.

The deviation measure Δ of these consecutive distances is then calculated for each run. An average of these deviations over 10 runs is calculated as the measure ($\overline{\Delta}$) for comparing different algorithms. Thus, it is clear that an algorithm having a smaller $\overline{\Delta}$ is better, in terms of its ability to widely spread solutions in the obtained front.

For all test problems and with NSGA-II, we use a population of size 100, a crossover probability of 0.8, a mutation probability of $1/n$ (where n is the number of variables). We run NSGA-II for 250 generations. The variables are treated as real numbers and the simulated binary crossover (SBX) [2] and the real-parameter mutation operator are used. For the (1+1)-PAES, we have used an archive size of 100 and depth of 4 [6]. A mutation probability of 0.01 is used. In order to make the comparisons fair, we have used 25,000 iterations in PAES, so that total number of function evaluations in NSGA-II and in PAES are the same.

Table 1 shows the deviation from an ideal (uniform) spread ($\overline{\Delta}$) and its variance in 10 independent runs obtained using NSGA-II and PAES. We show two columns for each test problem. The first column presents the $\overline{\Delta}$ value of 10 runs and the second column shows its variance. It is clear from the table that in all five test problems NSGA-II has found much smaller $\overline{\Delta}$, meaning that NSGA-II is able to find a distribution of solutions closer to a uniform distribution along the non-dominated front. The variance columns suggest that the obtained Δ values are consistent in all 10 runs.

Table 1. Comparison of mean and variance of deviation measure Δ obtained using NSGA-II and PAES

Algorithm	MOP2		MOP3		MOP4		EC4		EC6	
NSGA-II	0.361	0.00068	0.445	0.00043	0.387	0.00164	0.383	0.00099	0.365	0.01613
PAES	1.609	0.00671	1.341	0.00495	1.087	0.00687	1.563	0.05723	1.195	0.05151

In order to have a better understanding of how these algorithms are able to spread solutions over the non-dominated front, we present the entire non-dominated front found by NSGA-II and PAES in two of the above five test problems. Figures 2 and 3 show that NSGA-II is able to find a much better distribution than PAES on MOP4.

In EC4, converging to the global Pareto-optimal front is a difficult task. As reported elsewhere [11], SPEA converged to a front with $g = 4.0$ in at least one out of five different runs. With NSGA-II, we find a front with $g = 3.5$ in one out of five different

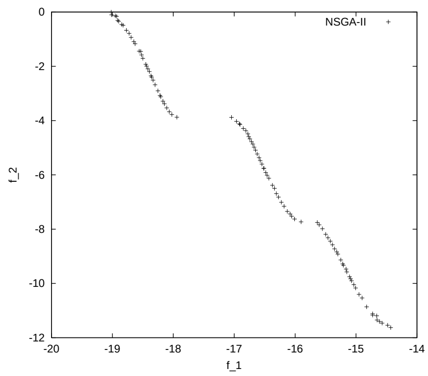

Fig. 2. Non-dominated solutions obtained using NSGA-II on MOP4.

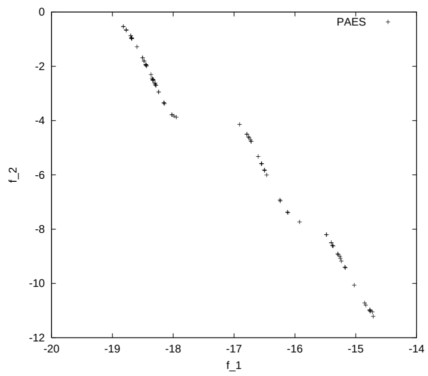

Fig. 3. Non-dominated solutions obtained using PAES on MOP4.

runs.

Figure 4 shows the non-dominated solutions obtained using NSGA-II and PAES for EC6. Once again, it is clear that the NSGA-II is able to better distribute its population along the obtained front than PAES. It is worth mentioning here that with similar number of function evaluations, SPEA, as reported in [11], had found only five different solutions in the non-dominated front.

5 Conclusions

In this paper, we have proposed a computationally fast elitist multi-objective evolutionary algorithm based on non-dominated sorting approach. On five difficult test problems borrowed from the literature, it has been found that the proposed NSGA-II outperforms PAES—another multi-objective EA with the explicit goal of preserving spread on the non-dominated front. With the properties of a fast non-dominated sorting procedure, an elitist strategy, and a parameterless approach, NSGA-II should find increasing attention and applications in the near future.

Acknowledgements

Authors acknowledge the support provided by All India Council for Technical Education, India during the course of this study.

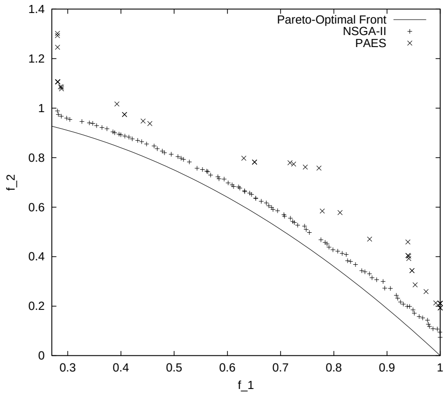

Fig. 4. Obtained non-dominated solutions with NSGA-II and PAES on EC6.

References

1. Deb, K. (1999) Multi-objective genetic algorithms: Problem difficulties and construction of test Functions. *Evolutionary Computation, 7*(3), 205–230.
2. Deb, K. and Agrawal, R. B. (1995) Simulated binary crossover for continuous search space. *Complex Systems, 9* 115–148.
3. Fonseca, C. M. and Fleming, P. J. (1993) Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, Morgan Kauffman, San Mateo, California.
4. Fonseca, C. M. and Fleming, P. J. (1998) Multiobjective optimization and multiple constraint handling with evolutionary algorithms–Part II: Application example. *IEEE Transactions on Systems, Man, and Cybernetics: Part A: Systems and Humans*. 38–47.
5. Horn, J. and Nafploitis, N., and Goldberg, D. E. (1994) A niched Pareto genetic algorithm for multi-objective optimization. In Michalewicz, Z., editor, *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 82–87, IEEE Service Center, Piscataway, New Jersey.
6. Knowles, J. and Corne, D. (1999) The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation. *Proceedings of the 1999 Congress on Evolutionary Computation*, Piscatway: New Jersey: IEEE Service Center, 98–105.
7. Mitra, K., Deb, K., and Gupta, S. K. (1998). Multiobjective dynamic optimization of an industrial Nylon 6 semibatch reactor using genetic algorithms. *Journal of Applied Polymer Science, 69*(1), 69–87.
8. Rudolph, G. (1999) Evolutionary search under partially ordered sets. Technical Report No. CI-67/99, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany.
9. Srinivas, N. and Deb, K. (1995) Multi-Objective function optimization using non-dominated sorting genetic algorithms, *Evolutionary Computation*, 2(3):221–248.
10. Weile, D. S., Michielssen, E., and Goldberg, D. E. (1996). Genetic algorithm design of Pareto-optimal broad band microwave absorbers. *IEEE Transactions on Electromagnetic Compatibility, 38*(4).

11. Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: Methods and applications. Doctoral thesis ETH NO. 13398, Zurich: Swiss Federal Institute of Technology (ETH), Aachen, Germany: Shaker Verlag.
12. Zitzler, E., Deb, K., and Thiele, L. (in press) Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation, 8*.
13. Zitzler, E. and Thiele, L. (1998) Multiobjective optimization using evolutionary algorithms—A comparative case study. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature, V*, pages 292–301, Springer, Berlin, Germany.