X·

THE MULTIOBJECTIVE GENETIC ALGORITHM APPLIED TO BENCHMARK PROBLEMS - AN ANALYSIS

R. C. PURSHOUSE

P. J. FLEMING

Department of Automatic Control and Systems Engineering
University of Sheffield
Sheffield, S1 3JD
UK

Research Report No. 796
August 2001

The Multiobjective Genetic Algorithm (MOGA) applied to Benchmark Problems – An Analysis

R. C. Purshouse and **P. J. Fleming**
Department of Automatic Control and Systems Engineering, University of Sheffield, Mappin Street, Sheffield, S1 3JD, UK.

Abstract

The multiobjective genetic algorithm (MOGA) has been applied to various real-world problems in a variety of fields, most prominently in control systems engineering, with considerable success. However, a recent empirical analysis of multiobjective evolutionary algorithms (MOEAs) has suggested that a MOGA-based algorithm performed poorly across a diverse set of two-objective test problems. In this report, it is shown that a conventional MOGA with standard settings can provide improved performance, but this still compares unfavourably to the best-performing contemporary MOEA, the Strength Pareto Evolutionary Algorithm (SPEA). The importance of the MOEA as a *framework* is stressed and, consequently, a real-coded MOGA for real-parameter multi-criterion problems is developed using modern guidelines for the design of evolutionary algorithms. This MOGA is shown to outperform all other published results across the benchmark problems. This does not suggest that MOGA is the 'best' MOEA, rather that a considered implementation of the methodology is required in order to reap full rewards. This study also questions the effectiveness of the traditional *fitness sharing* method of niching, with respect to the current set of multiobjective benchmark problems.

1 Introduction

Research into *evolutionary multicriterion optimisation* (EMO) has continued to escalate in popularity since the field's inception in the mid-1980s. Various multiobjective evolutionary algorithms (MOEAs) have been introduced and developed, with Pareto-based methods receiving the most attention. Fonseca and Fleming's [1993] *MOGA*, Horn and Nafpliotis' [1993] *NPGA*, and Srinivas and Deb's [1994] *NSGA* were the first three Pareto-based MOEAs. All three algorithms have proved popular in EMO applications, and have attracted continued development. Recently, Zitzler and Thiele's [1999] *SPEA* has established itself as another viable Pareto-based approach.

Predictably, the existence of alternative algorithms has instigated a degree of algorithmic competition into the EMO arena. Various multiobjective test suites have been devised in order to assess the ability of an MOEA in terms of various problem characteristics, such as non-convexity, multimodality, and non-uniformity. In a recent study by Zitzler *et al* [2000], an implementation of MOGA, labelled as *FFGA*, was found to perform poorly across a set of test problems with varying characteristics, relative to the performance of other MOEAs.

The motivation for this report is to explore the performance of MOGA on the Zitzler *et al* [2000] test suite in order to identify those aspects of the algorithm that are critical to success under various conditions. Throughout this report, the importance of the MOEA *methodology* is stressed rather than a tit-for-tat comparison of various algorithms.

After a general introduction to EMO and the MOGA methodology in Section 2, the performance of the FFGA, as described by Zitzler *et al* [2000], is validated using a MOGA with FFGA settings. In Section 4, the performance of the contemporary MOGA (defined in Fonseca and Fleming [1995, 1998]) is established and compared to that of the FFGA. Analysis of discrepancies is then simplified by making small changes to the FFGA settings and recording the results.

In Section 5, using the design approach intelligently expounded by Michalewicz and Fogel [2000], a MOGA is developed specifically for application to real-parameter problems. The performance of this MOGA on the five real-parameter problems in the Zitzler *et al* [2000] test suite is subsequently investigated in Section 6.

The report concludes in Section 7 by offering thoughts on the performance of MOGA, the benefits of empirical comparisons of different MOEAs, and the validity of the existing multiobjective test suite.

2 An introduction to MOGA

2.1 Overview

Multiobjective optimisation is the search for acceptable solutions to problems that incorporate multiple performance criteria. Often separate criteria, or objectives, are in competition with one another. In this case, a *trade-off* exists between the objectives, where improvement in one objective cannot be achieved without detriment to another. It is very rare for a multiobjective optimisation problem to admit a single optimal solution; rather a *family* of equally valid solutions will exist. This is illustrated in Figure 1.

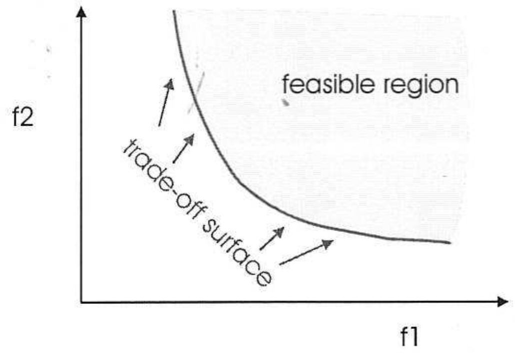

Figure 1: A trade-off between two competing objectives

Formally, and without loss of generality, multiobjective optimisation can be expressed as:

Minimise $\mathbf{f}(\mathbf{x})$,

where $\mathbf{f}(\mathbf{x}) = \{f_1(\mathbf{x}) .. f_n(\mathbf{x})\}$ is a vector of objective functions,

n is the number of objectives or criteria to be considered,

$\mathbf{x} = \{x_1 .. x_p\}$ is a vector of decision variables, and

p is the number of decision variables that comprise the complete solution.

In the absence of preference information, solutions to multiobjective problems are compared using the notion of *Pareto dominance.* A particular solution $\mathbf{x}$, with associated performance vector $\mathbf{u}$, is said to dominate, or be better than, another solution $\mathbf{y}$ with performance vector $\mathbf{v}$ ($\mathbf{x} \prec \mathbf{y}$) if the former performs at least as well as the latter across all objectives, and exhibits superior performance in at least one objective. The formal definition is given in Equation 1.

$$\mathbf{u} \prec \mathbf{v} \text{ iff } \left[\forall i \in \{1,\dots,n\}, u_i \le v_i\right] \cap \left[\exists i \in \{1,\dots,n\} : u_i < v_i\right] \quad (1)$$

$$\mathbf{u} \prec \mathbf{v} \Leftrightarrow \mathbf{x} \prec \mathbf{y}$$

where u_i / v_i is the ith criterion value of the performance vector $\mathbf{u}$ / $\mathbf{v}$.

A solution is said to be *Pareto optimal* if it is not dominated by any other possible solution, as described by Equation 2. The *Pareto-front* is the set of points in criterion-space that correspond to the Pareto-optimal solutions. These concepts are illustrated in Figure 2. Without *a priori* or progressive preference articulation, a multiobjective search engine will generally aim to discover a family of solutions that provide a good representation of the Pareto front.

$$\mathbf{x} \in X_{PO} \text{ iff } \nexists \mathbf{y} \in U : \mathbf{y} \prec \mathbf{x} \quad (2)$$

where X_{PO} is the set of Pareto optimal solutions, and

U is the set of all feasible solutions.

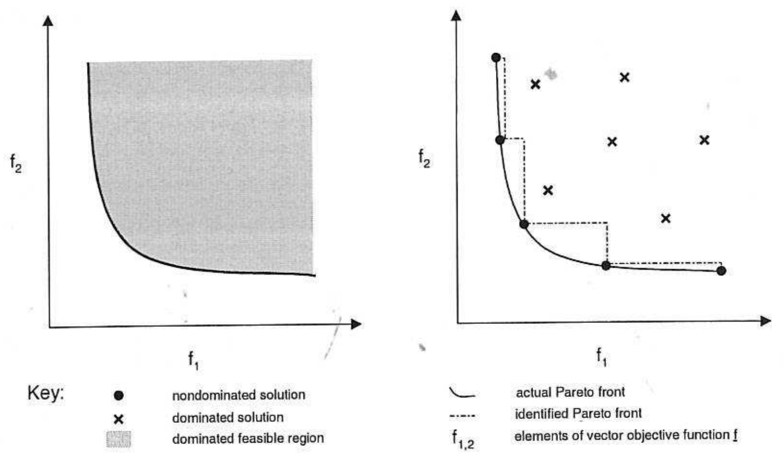

Figure 2: Pareto optimality

Genetic algorithms were first proposed as multiobjective optimisers by Schaffer [1984]. However, the first *Pareto-based* multiobjective evolutionary algorithm (MOEA) to be published was the *multiobjective genetic algorithm* (MOGA) developed by Fonseca and Fleming [1993]. Genetic algorithms are suitable search engines for multiobjective problems primarily because of their population-based approach. An MOEA is capable of supporting diverse, simultaneous, solutions in the search environment. A carefully designed GA is robust in the face of ill-behaved cost landscapes featuring attributes such as multimodality and discontinuity. Furthermore, the GA methodology offers a flexible choice of decision variables and objective specifications. Refer to Veldhuizen and Lamont [2000] and Coello [1999] for recent surveys of MOEA research. A general schematic of the MOGA is shown in Figure 3.

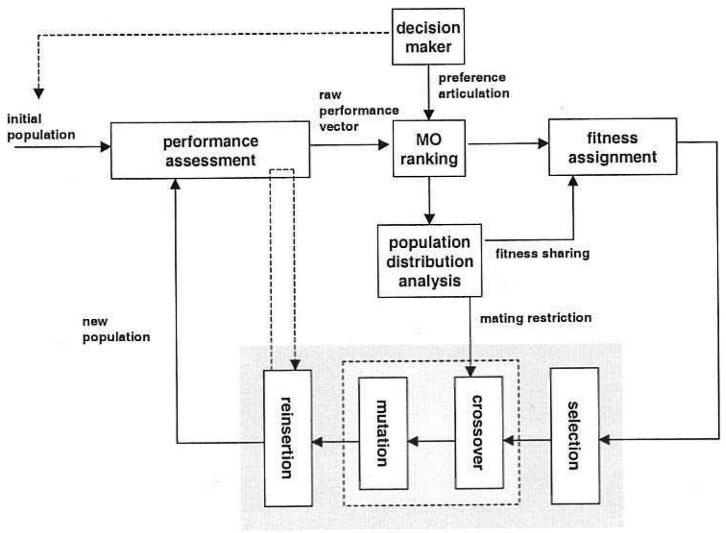

Figure 3: MOGA schematic

The MOGA framework can be seen to incorporate all the elements of the standard, single objective, genetic algorithm. A population of potential solutions is instantiated, then assessed and manipulated over a number of iterations in order to obtain a good solution or set of solutions. Performance assessment, selection, genetic operators (such as crossover and mutation), and reinsertion phases are functionally, in a general sense, the same for the MOGA as for the standard GA. Population distribution analysis, in which a measure of the density of the population is made, has also been applied in the single objective case to cater for multimodal cost landscapes. The results of this analysis are used in *niching* and *mating restriction* schemes. These schemes can take on a somewhat different meaning for MOEAs, as discussed in Sections 2.3 and 2.4. Multiobjective ranking, which impacts primarily on fitness assignment, is the key discrepancy between the MOGA and a standard GA. This aspect receives further attention in Section 2.2.

Interaction with a decision-maker (DM), or group of decision-makers, is made explicit in Figure 3. The DM may choose to introduce *a priori* information into the initial population (at the very least, this would include appropriate limits on decision variables), as is sometimes the case in standard GA applications. With the MOGA, the DM can also seek to influence the search whilst it is in progress by expressing preference for particular solutions or, more generally, the likely attributes of a good solution. This is discussed further in the following sub-section.

2.2 Multiobjective ranking

The essential difference between a MOGA and a single objective GA is the method by which fitness is assigned to potential solutions. Each solution will have a vector describing its performance across the set of criteria. This vector must be transformed into a scalar fitness value for the purposes of the GA. This process is achieved by ranking the population of solutions relative to each other, and then assigning fitness based on rank. Individual solutions are compared in terms of Pareto dominance. This notion was introduced into the field of genetic algorithms by Goldberg [1989]. In Goldberg's formulation, the population at each generation is searched for nondominated solutions. These are assigned rank 0 and are then temporarily removed from the population. Nondominated solutions are then identified in the remaining population, and these are assigned rank 1 and removed from contention. The process continues until all individuals have been ranked. In effect, this process creates a series of nondominated fronts. Srinivas and Deb [1994] adopted this approach for their *Nondominated Sorting Genetic Algorithm* (NSGA).

MOGA uses a variation of Goldberg's proposition in order to determine ranks. Each individual is assigned a rank based on the number of individuals by which it is dominated. This is illustrated in Figure 4.

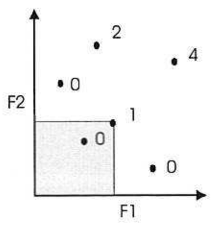

Figure 4: Multiobjective ranking

It should be noted that, in the other early Pareto-based approach, Horn and Nafpliotis [1993] avoided a global ranking scheme through the implementation of Pareto domination tournaments. In this approach, binary tournaments are conducted in which the winner is deemed to be the individual that is dominated by fewer solutions chosen from a random sample of the population.

In the absence of preference information, Pareto dominance is used to discriminate between two competing solutions. However, by involving a decision-maker (DM) in the search, other factors can be used to determine superiority. Fonseca and Fleming [1998] introduced a *preferability* operator, which discriminates between solutions on the basis of which is preferred by the DM.

In Fonseca and Fleming's scheme, the DM can set goal levels and priorities for each of the objectives. These can be refined as the search progresses. This information feeds into the preferability operator, which is used to rank solutions in a similar fashion to the standard Pareto-based approach. Each potential solution is given a rank based on how many other solutions are preferred to it. The mathematical definition of the operator is quite involved and is omitted from this discussion. For a complete definition refer to [Fonseca and Fleming, 1998].

The preferability operator can be seen as a unification of several popular preference articulation schemes adopted in the wider operational research community. Pareto optimality, the *lexicographic* method, *goal programming*, *constraint satisfaction*, and *constrained optimisation* can all be described by special cases of the preferability operator.

2.3 Niching

2.3.1 Overview

Niching refers to the deliberate formation of clusters of individuals from the wider GA population in either solution-space or criterion-space. Since multiobjective problems contain a distributed family of Pareto-optimal solutions, niching is undertaken in order to achieve a good spread of discovered solutions to present to the decision-maker. Without niching, the population may converge to a localised region, in a phenomenon known as *genetic drift*. Furthermore, given that the family of Pareto-optimal solutions may be large, relative to the size of the GA's population, niching can prevent the GA from being swamped by solutions all with identical fitness.

Various forms of niching have been implemented, both for single objective and multiobjective problems [Goldberg, 1989]. *Fitness sharing* is the most popular technique for achieving niching, in which the fitness assignment mechanism is modified to account for the local population distribution around each individual, as indicated in Figure 3. Two approaches to fitness sharing are discussed in the following sub-sections.

There has been some debate as to whether sharing should be performed in either the solution-space (the parameters manipulated by the GA) or the criterion-space (the results corresponding to the chosen parameters). Sharing in solution-space should provide a good distribution of alternative solutions, but this approach cannot guarantee a good distribution in criterion-space. The opposite is true for criterion-space sharing. Essentially, as suggested by Horn *et al* [1994], sharing should be performed in the space where a good distribution is deemed the most important. Of course, there is no reason why sharing cannot be performed in both domains. Note that a good distribution across the Pareto-front (which exists in criterion-space) is important in order to understand the trade-offs between the various objectives.

When multiobjective ranking is used as a means of fitness assignment, fitness sharing is often only applied to individuals of equal rank (the connection is made explicit in Figure 3). This is because genetic drift becomes most apparent when the fitnesses of different individuals are equivalent (which implies that the ranks are identical).

2.3.2 Goldberg and Richardson's approach

Fitness sharing was introduced by Goldberg and Richardson [1987] in order to encourage the formation of sub-populations on the various peaks of a multimodal cost function, thus avoiding sub-optimal convergence. This technique was originally devised for single objective problems but has found use in the multiobjective domain.

In essence, the fitness of each solution is reduced by a factor, known as the *niche count*, which depends on its 'closeness' to other solutions in the population. Niche counts are found by comparing a particular individual with all individuals in the population (including itself) on a pair-wise basis. The niche count for an individual is taken as the sum of all pair-wise counts. The equation used to calculate a single element of the niche count is shown in Equation 3.

$$sh(d) = \begin{cases} 1 - \left(\dfrac{d}{\sigma_{share}} \right)^{\alpha}, & d < \sigma_{share} \\ 0, & \text{otherwise}. \end{cases} \tag{3}$$

where sh	is the contribution to the individual's share count,
d	is the distance between two individuals, measured over some metric (normally L-2),
σ_{share}	is the niche size, and
α	is a shaping parameter.

Note that 'closeness' requires a definition of distance, prior to defining what is regarded as a close distance. Euclidean distance is a common choice for real-parameter functions. Having defined the unit of distance, closeness – embodied by the *niche size* - must then be defined. This is not an easy task, and requires a subjective judgement to be made, given the size of the search space. Deb and Goldberg [1989] suggested a method of calculating the niche size, originally applied to solution-space niching for single objective problems (see Equation 4). Fonseca and Fleming [1993] suggested an alternative method for calculating the niche size in criterion-space for multiobjective problems, but in the case of MOGA this has largely been superseded in favour of Epanechnikov fitness sharing (described in the following sub-section).

$$\sigma_{share} = \frac{\sqrt{\sum_{k=1}^{p} \left(x_{k,max} - x_{k,min} \right)^2}}{2\sqrt[p]{q}} \tag{4}$$

where p	is the number of decision variables within the solution **x**,
$x_{k,max/min}$	is the maximum / minimum value for the *k*th decision variable, and
q	is the required number of niches.

Goldberg and Richardson's power law sharing functions (Equation 3) have proved to be a popular choice for niche formation in both single objective and multiobjective spaces. However, choice of the niche size parameter is often difficult, and can be crucial to the success of the algorithm. This limitation has lead to increased research into alternatives to the standard fitness sharing algorithm, especially techniques that do not require explicit setting of the niche size.

2.3.3 Epanechnikov niching

In light of the above concerns, Fonseca and Fleming [1995] proposed an alternative fitness sharing algorithm. They reinterpreted the share count as the estimation of the population density at the points defined by each individual (in either solution- or criterion-space). This estimate is achieved using kernel density techniques. The critical benefit of this approach is that statisticians have developed techniques to determine good values for the parameter analogous to niche size [Silverman, 1986].

Fitness sharing	Kernel density estimation
Sharing function	Kernel function
Niche size, σ_{share}	Smoothing parameter, h
Niche count, $\sum sh$	Density estimate, $\sum K_e$

Table 1: The analogy between sharing and kernel estimation

Fonseca and Fleming used the *Epanechnikov* kernel as a sharing function. The kernel is described by Equation 5. Note the resemblance to the standard sharing function shown in Equation 3. The analogy is explicitly made in Table 1.

$$K_e(d/h) = \begin{cases} \frac{1}{2}c_p^{-1}(p+2)\left[1-(d/h)^2\right] & \text{if } d/h < 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

where	p	is the number of decision variables in the decision vector,
	c_p	is the volume of the unit p-dimensional sphere,
	d/h	is the normalised L_2 distance between individuals, and
	h	is the kernel smoothing parameter.

The kernel smoothing parameter, h, is directly analogous to Goldberg and Richardson's niche size parameter, σ_{share}. A good value for this parameter can be found using Equation 6 [Silverman, 1986]. This value is approximately optimal in the least-mean-integrated-squared-error sense if the population follows a multivariate normal distribution and has identity covariance matrix. For a population with arbitrary covariance matrix, S, the population should be transformed through multiplication by a matrix R, where $RR^T = S^{-1}$.

$$h = \left[8c_p^{-1}(p+4)\left(2\sqrt{\pi}\right)^p / N\right]^{1/(p+4)} \tag{6}$$

where N is the population size, and other parameters are defined as before.

It should be noted that the Epanechnikov-method has been adopted as the sharing method of choice in current MOGA applications [Chipperfield and Fleming, 1996; Griffin *et al*, 2000; Schroder *et al* 2001].

2.4 Mating restriction

When niching was implemented in single objective problems, Deb and Goldberg [1989] noted that recombination between chromosomes in different niches often produced unsuccessful offspring, known as *lethals*. This led to a degradation of GA performance. In order to remedy the problem, Deb and Goldberg decided to restrict mating over some distance metric, a technique previously considered for niche formation. The chosen distance measure was Euclidean distance in phenotypic solution-space. This follows logically from the choice of distance metric for fitness sharing, as described in Section 2.3. The maximum distance at which mating was permitted was chosen to be the same value as for the niche size used in fitness sharing. This makes some sense, since the niche size should be directly related to the spacing between niches. This convention has been adopted in most niching schemes that use mating restriction.

Mating restriction, as developed by Deb and Goldberg, has been directly applied to MOGA, as indicated in Figure 3. However, Fonseca and Fleming [1993] noted that the effectiveness of the technique may be diminished for multiobjective problems.

2.5 Summary

In this section, a methodology for solving multiobjective problems was developed using an evolutionary computing approach. It is stressed that, in very much the same way as the GA itself, MOGA represents a *framework* for problem-solving rather than a panacea-like tool or piece of software. There are two fundamental aspects to MOGA, namely:

- its **population-based** nature, and
- its **Pareto-based** method of comparing solutions that facilitates *a posteriori* and progressive **preference articulation**.

MOGA is proving to be an increasingly popular technique across a growing range of applications. Since its conception, it has found great interest amongst control and systems engineers. Some sample applications are listed in Table 2 below.

Application	Reference
Radiotherapy treatment planning	Haas *et al* [1997]
Supersonic wing shape optimisation	Obayashi *et al* [2000]
H-infinity design of a maglev vehicle	Dakev *et al* [1997]
Identification of NARMAX models	Rodríguez-Vázquez *et al* [1997]
On-line controller tuning (prior to usage)	Schroder *et al* [2001]

Table 2: Sample MOGA applications

3 Validation of published results

3.1 Introduction

In their comparison of multiobjective evolutionary algorithms, Zitzler *et al* [2000] report that their implementation of MOGA, based on the paper by Fonseca and Fleming [1993], performs poorly in comparison to other MOEAs. Zitzler *et al* refer to this implementation of MOGA as the *FFGA*. In this section, the published results are validated on Ziztler *et al*'s test functions, derived from those developed by Deb [1999], using a MOGA with the same specification as the FFGA.

3.2 Baseline MOGA – the 'FFGA'

The MOGA used to validate the FFGA results was developed to the specifications described in Table 3. These settings were defined in [Zitzler *et al*, 2000].

MOEA parameter	Setting
General GA	
Population size	100
Total generations	250
Coding	Binary, 30 bits per decision variable (except where varied, as shown in Zitzler *et al* [2000]).
Selection	Stochastic universal sampling [Baker, 1987]
Recombination	Single-point binary crossover, probability = 0.8
Mutation	Element-wise bit-flipping, probability = 0.01
Generational gap	Zero
Random injection	Zero random chromosomes per generation
Elitism	None
Multiobjective GA	
Fitness assignment	Fonseca and Fleming's [1993] multiobjective ranking (see Section 2.2). Transformation from rank to fitness using linear fitness assignment with rank-wise averaging.
External population	Off-line storage of nondominated solutions
Niching	
Fitness sharing	Rank-wise Goldberg and Richardson [1987] fitness sharing in criterion-space. Parameters: alpha = 2, niche size = 0.48862.
Mating restriction	None

Table 3: MOGA emulation of the FFGA

3.3 Results

A summary of the functions in the test suite is provided in Table 4. For a full description of the functions, including the equations, refer to Zitzler *et al* [2000].

Test number	Features
ZDT-T1	Convex Pareto front.
ZDT-T2	Non-convex Pareto front.
ZDT-T3	The Pareto front consists of several non-contiguous convex parts.
ZDT-T4	Contains 21^9 local fronts, with a single contiguous global Pareto-optimal front.
ZDT-T5	Deceptive problem.
ZDT-T6	Non-uniform distribution of solutions along a non-convex Pareto front

Table 4: Multiobjective test suite

The investigation conducted here mirrors that in Zitzler *et al* [2000]. Each configuration of MOGA is applied thirty times to each test problem. The results across the first five replications are amalgamated, and the nondominated solutions are extracted, for each problem in the test suite. This sub-set of results, shown in Figures 5 through 10, is used for visual analysis. The baseline MOGA (bMOGA) results (depicted by O) are compared with Zitzler *et al*'s published FFGA results (◇) and the results for the *Strength Pareto Evolutionary Algorithm* (SPEA) [Zitzler and Thiele, 1999], which achieved the best results across all test problems in Zitzler *et al*'s study (□). The global Pareto front is indicated by the solid curve.

As illustrated in Figure 5, the baseline MOGA offers a significant improvement over the reported FFGA results for ZDT-T1, both in terms of closeness to the global front and distribution across the front. However, SPEA clearly outperforms the bMOGA on both these aspects.

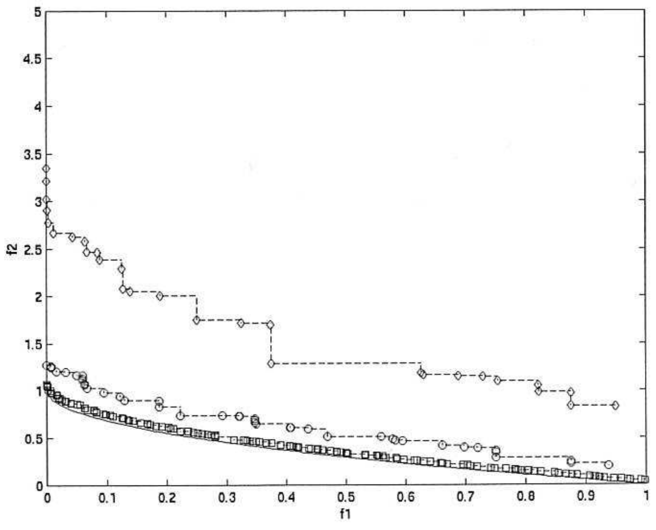

Figure 5: Results for ZDT test function 1 (◇– FFGA; O – bMOGA, □ – SPEA)

Results for ZDT-T2 are shown in Figure 6. The ordering of the MOEAs in terms of performance is clearly the same as for ZDT-T1. Interestingly, the relative degrees of performance are very similar for both problems. Note that all three algorithms struggle to provide good coverage of the region of the front where objective f_1 tends towards zero.

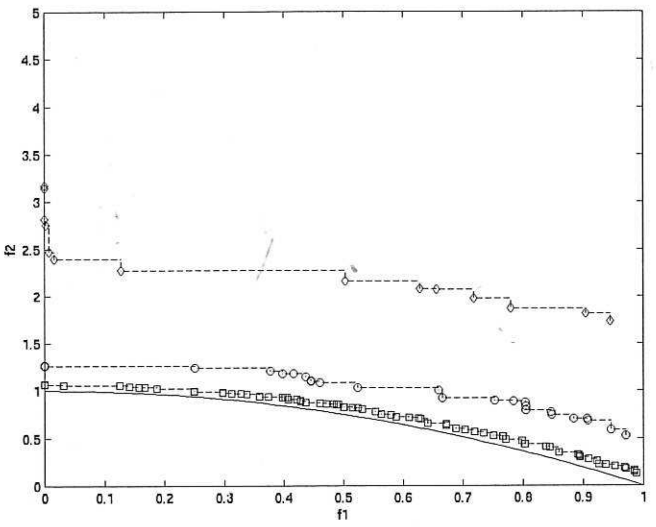

Figure 6: Results for ZDT test function 2 (◇– FFGA; O – bMOGA, □ – SPEA)

Again, for ZDT-T3, the relative degree of attainment is closely matched to that observed in the previous tests, as shown in Figure 7. The baseline MOGA clearly exhibits a superior distribution of Pareto optimal solutions than its FFGA equivalent.

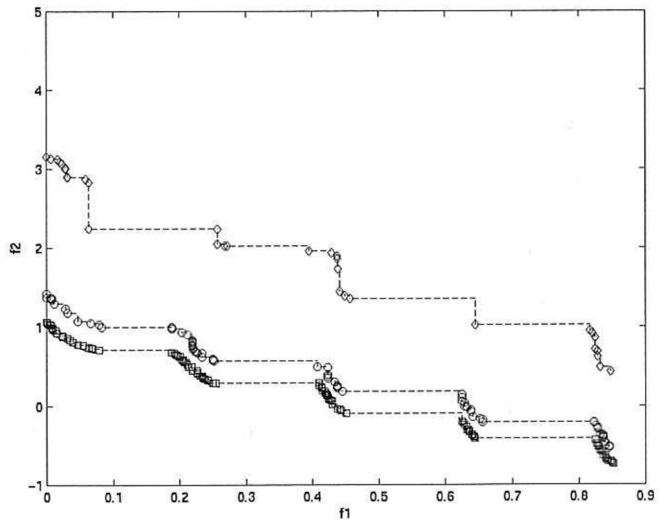

Figure 7: Results for ZDT test function 3 (◇– FFGA; O – bMOGA, □ – SPEA)

The baseline MOGA and the FFGA both struggle to provide acceptable results for ZDT-T4. This is illustrated in Figure 8. bMOGA is able to find solutions significantly closer to the global front, but neither algorithm is capable of finding a good distribution of solutions. SPEA is evidently able to produce a suitable distribution of locally nondominated points, but the closeness to the global front is similar to that of bMOGA. None of the MOEAs tested by Zitzler *et al* [2000] were capable of identifying the true Pareto front for this multi-fronted problem.

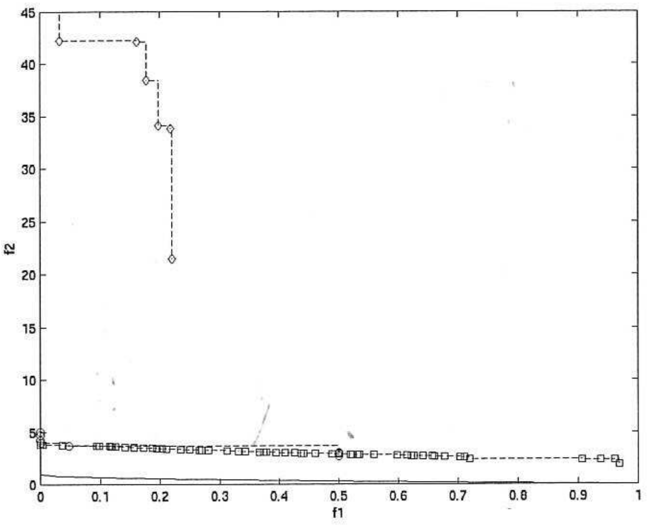

Figure 8: Results for ZDT test function 4 (◇– FFGA; O – bMOGA, □ – SPEA)

According to the results presented by Zitzler *et al* [2000], ZDT-T5 is the only test function for which the performance of different MOEAs seems to converge somewhat. Indeed, bMOGA produces results closer to the front than SPEA, although the latter algorithm provides a superior distribution at large values of objective f_1. Note that the FFGA appears to struggle with the central region of the front (where, if both objectives are of similar importance, attractive compromise solutions may reside) but that this behaviour was not replicated by the bMOGA. The ZDT-T5 results are displayed in Figure 9.

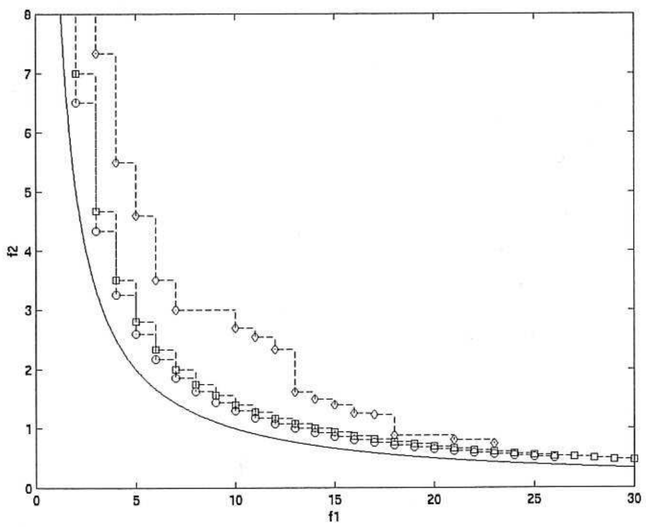

Figure 9: Results for ZDT test function 5 (◇– FFGA; O – bMOGA, □ – SPEA)

Baseline MOGA and FFGA are uncompetitive with SPEA on ZDT-T6 (see Figure 10). The latter algorithm is capable of accurately finding the global front, although distribution across the front is somewhat sparse. bMOGA has been able to identify the extremes of the front (with low accuracy), but the FFGA has discovered the non-convex central region (although, again, this is some distance away from the global front).

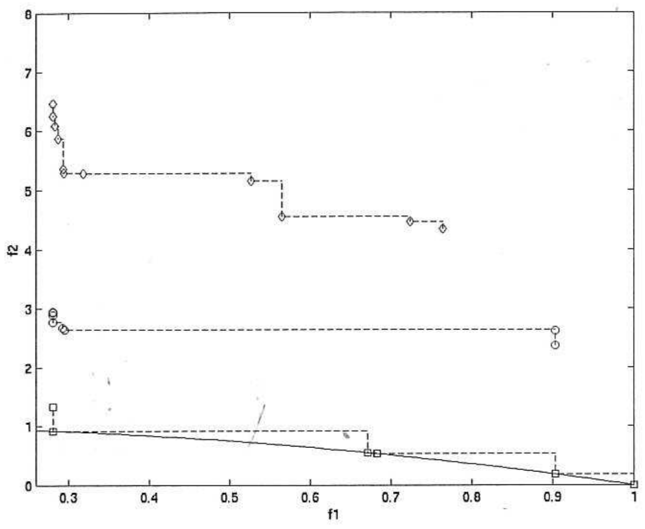

Figure 10: Results for ZDT test function 6 (◇– FFGA; ○ – bMOGA, □ – SPEA)

The performance metric described by Zitzler *et al* [2000] can also be used as a basis for comparing different MOEAs. A coverage metric is used, which expresses the proportion of solutions found by one MOEA that are equal to or dominate ($\preceq$) the solutions found by a second MOEA. This is described by Equation 7.

$$C(A,B) = \frac{\left|\{\mathbf{b} \in B; \exists \mathbf{a} \in A : \mathbf{a} \preceq \mathbf{b}\}\right|}{|B|} \tag{7}$$

where A, B are each a set of nondominated criterion vectors, and

a, b are particular criterion-vectors from sets A and B respectively.

A measure of coverage is obtained for each pair-wise run of two MOEAs. The results for all thirty runs are summarised by the box plots in Figures 11 and 12. Each column in the figure represents a box plot of the results for a particular test function. Each box plot encodes the results of the thirty coverage comparisons. The thick, unbroken, horizontal line indicates the median level of coverage of the thirty results. The box itself represents 50% of the distribution, where the upper and lower ends of the box represent the upper and lower quartiles respectively. The appendages to the box indicate the shape and spread of the tails of the distribution. Outliers are represented by crosses. Refer to Cleveland [1993] for a good introduction to box plots and other methods for visualising data.

A comparison of bMOGA and FFGA is shown in Figure 11. With the exception of a single outlier, bMOGA comprehensively outperforms its FFGA equivalent across all test functions using the coverage performance metric. Coverage of bMOGA fronts by FFGA-discovered fronts is zero.

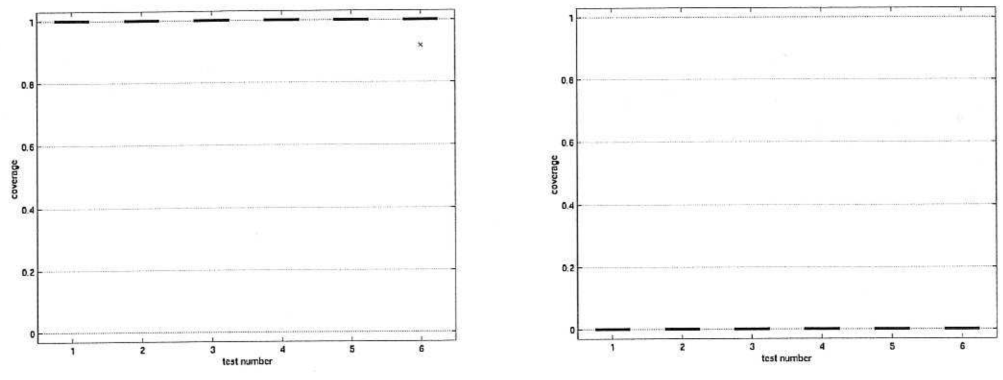

Figure 11: Performance comparison - (a) *C(bMOGA, FFGA)*; (b) *C(FFGA, bMOGA)*

When comparing bMOGA and SPEA (see Figure 12), the latter algorithm completely surpasses the former across the first three test functions, apart from a small number of outliers on ZDT-T3 where coverage is less than 100%. Some variability exists in the results for ZDT-T4, perhaps suggesting that bMOGA can provide superior accuracy to SPEA on occasions but that the algorithm is hampered by poor distribution (clearly indicated by the visual presentation in Figure 8). There is also significant variability in the results for ZDT-T5. When considering the median level of coverage in isolation, it could be strongly argued that bMOGA has produced the better results. This validates the outcome shown in Figure 9. There is also some variation in the results for ZDT-T6, although SPEA clearly produces solutions that cover many of those found by bMOGA. There is zero coverage of SPEA solutions by bMOGA solutions.

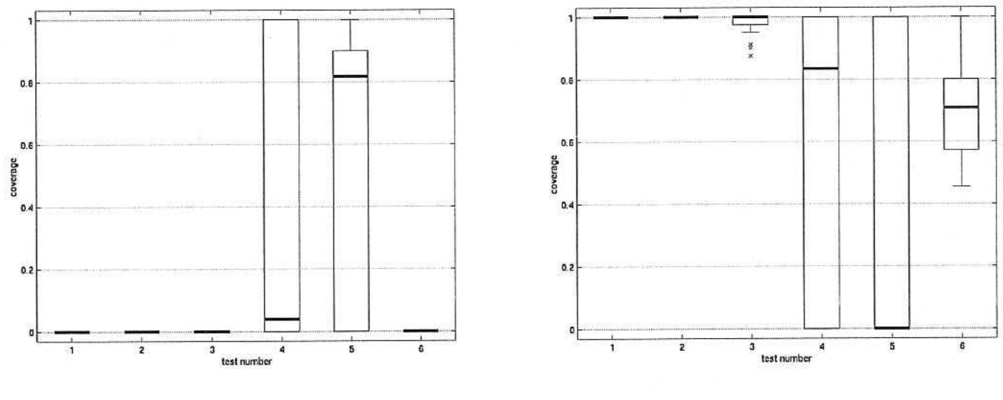

Figure 12: Performance comparison - (a) *C(bMOGA, SPEA)*; (b) *C(SPEA, bMOGA)*

3.4 Conclusion

The validated results suggest that the implementation of bMOGA described in this report does not perform as poorly as Zitzler *et al*'s FFGA implementation. Note that both implementations are based on Fonseca and Fleming's 1993 paper. This difference in results can perhaps be explained by differing interpretations of that paper.

However, it is clear that both the bMOGA and the FFGA fail to produce results to match those of the SPEA. In the next section, modifications to the baseline MOGA are sought that will improve the performance of the algorithm. The current MOGA algorithm, that described in [Fonseca and Fleming, 1995] and [Fonseca and Fleming, 1998], is also applied to the test functions. The eventual aim is to determine a set of recommended settings for MOGA that apply to the test suite and, thus, to any inferred classes of problems.

4 Analysis and improvement of MOGA performance

4.1 Introduction

Results in the previous section indicated that a MOGA with the settings described in Zitzler *et al* [2000] did not perform as poorly as the said authors' FFGA. However, performance was still distinctly inferior to that of other MOEAs, in particular the SPEA. Contemporary benchmark settings for MOGA (see Fonseca and Fleming [1995, 1998], Chipperfield and Fleming [1996], and Schroder *et al* [2001]) differ somewhat from the settings described by Zitzler *et al.* Hence, in this section, the performance of the contemporary MOGA implementation is tested and assessed on the ZDT test functions. Subsequently, various simple adjustments to the FFGA-esque baseline MOGA (bMOGA) are considered in order to identify the particular aspects of the algorithm to which performance is sensitive.

4.2 Contemporary MOGA

Settings for the *de facto* MOGA, as used today, are shown in Table 5. The bMOGA settings are also included for comparison purposes. Whilst it is stressed that MOGA is a methodology rather than a specific algorithm, these settings can be regarded as a benchmark from which other MOGAs will vary. Note that population size, number of generations, and decision variable resolution are chosen in line with Zitzler *et al* [2000]. Differences between bMOGA and MOGA are highlighted in **bold** typeface.

MOEA parameter	bMOGA setting	MOGA setting
General GA		
Population size	100	100
Total generations	250	250
Coding	Binary, 30 bits per decision variable (except where varied, as shown in Zitzler *et al* [2000]).	**Gray**, 30 bits per decision variable (except where varied, as shown in Zitzler *et al* [2000]).
Selection	Stochastic universal sampling [Baker, 1987]	Stochastic universal sampling [Baker, 1987]
Recombination	Single-point binary crossover, probability = 0.8	Single-point binary crossover, probability = **0.7**
Mutation	Element-wise bit-flipping, probability = 0.01	Element-wise bit-flipping, **expectation of 1 bit per chromosome**
Generational gap	Zero	Zero
Random injection	Zero random chromosomes per generation	**2** random chromosomes per generation
Elitism	None	None

Table 5: Benchmark MOGA

MOEA parameter	bMOGA setting	MOGA setting
Multiobjective GA		
Fitness assignment	Fonseca and Fleming's [1993] multiobjective ranking (see Section 2.2). Transformation from rank to fitness using linear fitness assignment with rank-wise averaging.	Fonseca and Fleming's [1993] multiobjective ranking (see Section 2.2). Transformation from rank to fitness using linear fitness assignment with rank-wise averaging.
External population	Off-line storage of nondominated solutions	Off-line storage of nondominated solutions
Niching		
Fitness sharing	Rank-wise Goldberg and Richardson [1987] fitness sharing in criterion-space. Parameters: alpha = 2, niche size = 0.48862.	(Parameter-less) **Epanechnikov fitness sharing** [Fonseca and Fleming, 1995]. Implemented in criterion-space.
Mating restriction	None	**Mating restriction implemented**: distance set to the niche size parameter found by the Epanechikov fitness sharing algorithm.

Table 5: Benchmark MOGA (continued)

Real-world applications of MOGA have tended to vary in their choice of sharing domain. Some applications, such as [Fonseca and Fleming, 1998], have implemented criterion-based sharing, whilst others, for example [Chipperfield and Fleming, 1996], have preferred a solution-based approach. In this empirical study of the contemporary MOGA, criterion-based sharing is initially considered in order to facilitate an easier comparison with the results in [Zitzler *et al*, 2000].

Results for the contemporary MOGA with benchmark settings are shown in Figures 13 to 18. The performance of the algorithm is compared to that of the baseline MOGA (see Section 3.2) and the best performing algorithm identified in Zitzler *et al*'s [2000] study, the SPEA. The first six figures show a unification of the first five runs for each algorithm on each test problem. From these plots, it is possible to obtain a feel for the accuracy and distribution of each identified Pareto front. From a statistical viewpoint, given the small sample size, confidence is not high. The coverage metric (Equation 7), which takes into account the results for all thirty runs, provides a mechanism for increased confidence, but does not provide the ease of comparison of the more informal visual presentation. Coverage results are displayed in Figures 19 to 22.

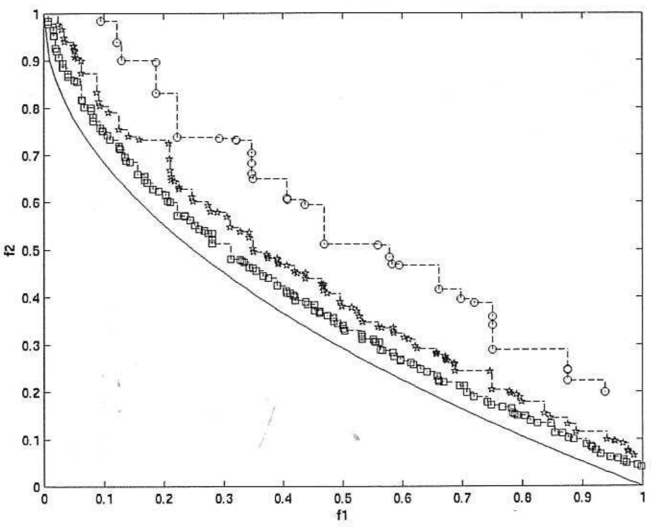

Figure 13: ZDT-T1 results (O – bMOGA, ☆ – MOGA, □ – SPEA)

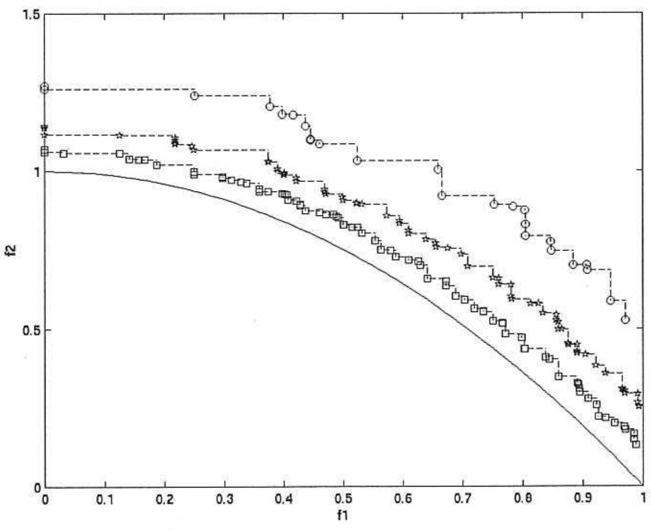

Figure 14: ZDT-T2 results (O – bMOGA, ☆ – MOGA, □ – SPEA)

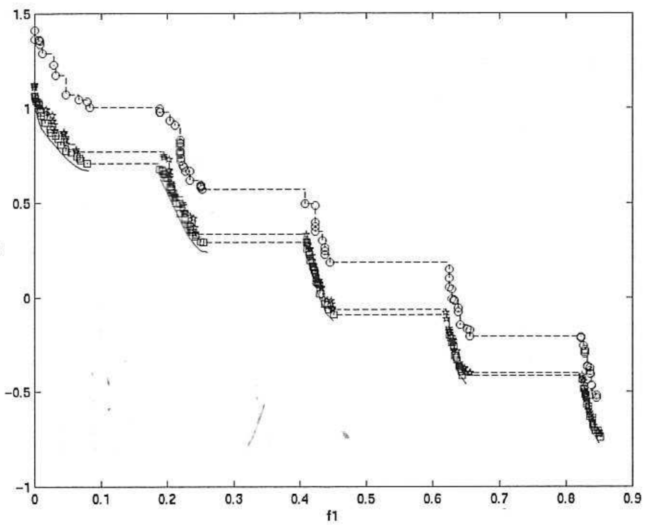

Figure 15: ZDT-T3 results (O – bMOGA, ☆ – MOGA, □ – SPEA)

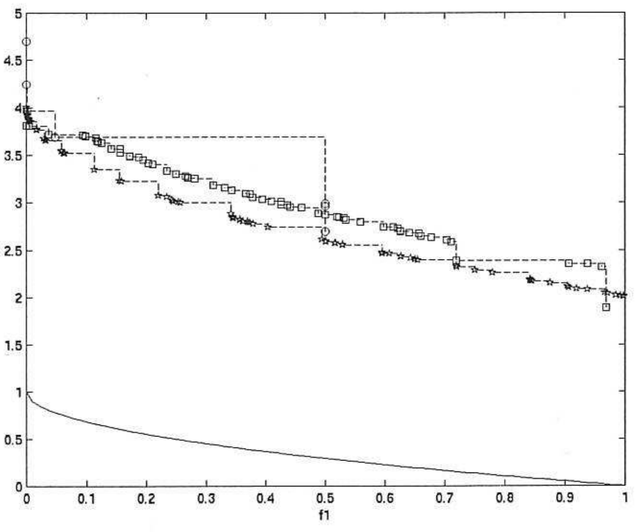

Figure 16: ZDT-T4 results (O – bMOGA, ☆ – MOGA, □ – SPEA)

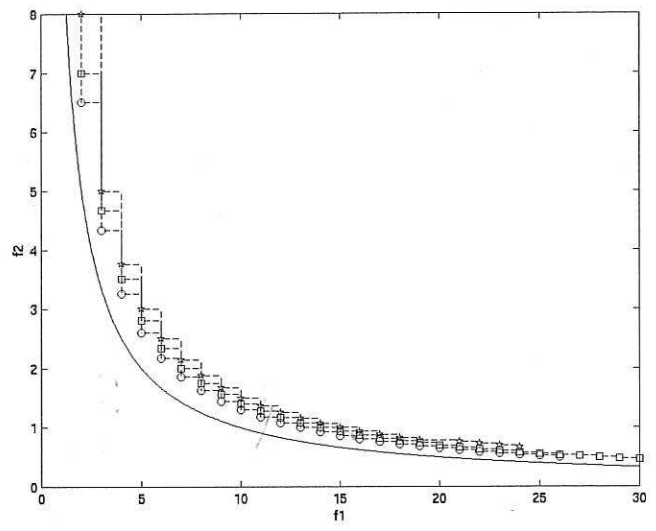

Figure 17: ZDT-T5 results (O – bMOGA, ☆ – MOGA, □ – SPEA)

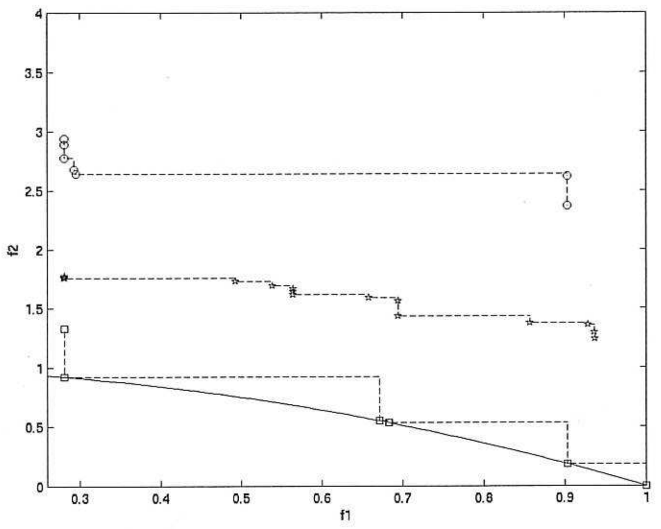

Figure 18: ZDT-T6 results (O – bMOGA, ☆ – MOGA, □ – SPEA)

It is clearly evident that the contemporary MOGA with benchmark settings performs better than the baseline MOGA (bMOGA) across all the test functions, with the possible exception of ZDT-T5. The difference in performance is apparent in the coverage box plots of Figure 19, in addition to the visual presentation above. This distinction is particularly marked for ZDT-T4 (Figure 16) where, unlike the bMOGA, the contemporary MOGA is capable of obtaining a good distribution of solutions across the Pareto front. On ZDT-T5, the median levels of coverage are both high, indicating that the discovered fronts are similar.

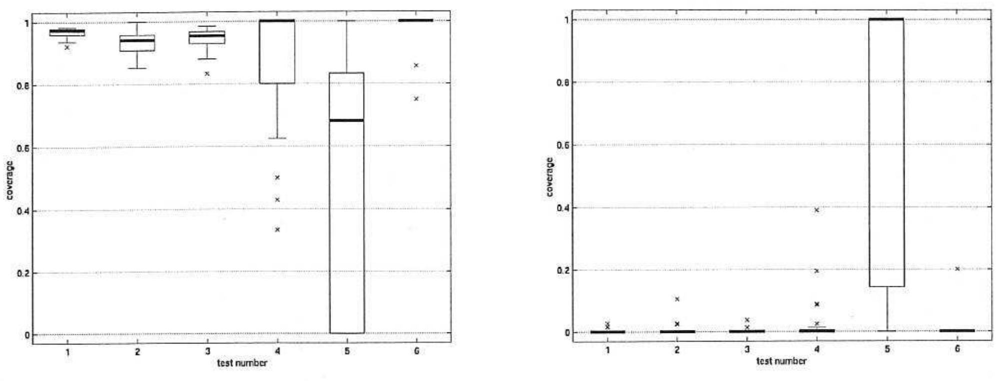

Figure 19: Coverage metric – (a) *C(MOGA, bMOGA)*; (b) *C(bMOGA, MOGA)*

It will be noted that the performance of MOGA is still inferior to that of the SPEA on test functions 1, 2, 3, and 6. However, MOGA provides a significant improvement over the SPEA on test function 4. The results in Figure 16 suggest that MOGA outperforms the SPEA in terms of both closeness to the global front and distribution across the front. This is borne out by the coverage metric box plots in Figure 20 where, despite a degree of variability in the distribution, the median level of coverage of SPEA by MOGA is 100%, whereas the reverse comparison is 0%. On ZDT-T5, MOGA is able to produce results of similar accuracy to SPEA, but the latter algorithm provides a better distribution as the value of objective f_1 increases. Judging by the results in Figure 18, MOGA can produce a superior distribution along the trade-off surface to SPEA. However, since the SPEA results are closer to the actual front, the MOGA results are largely covered (see Figure 20).

Note that the distribution obtained by MOGA on ZDT-T2 is still poor as the value for objective f_1 approaches zero, relative to the remainder of the front.

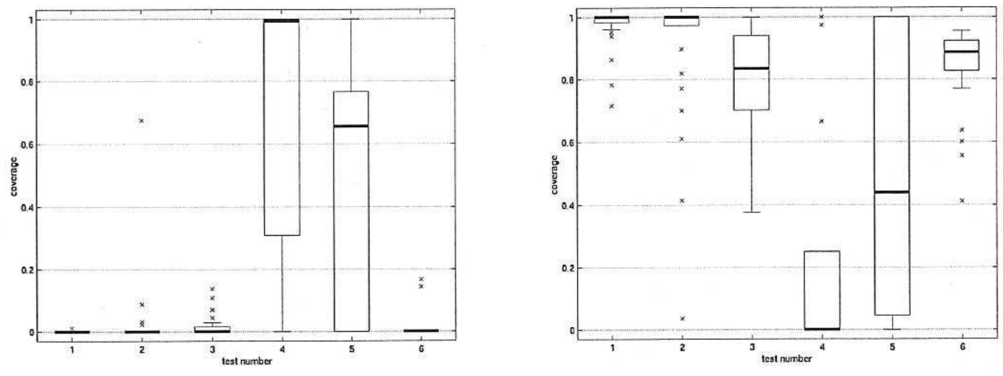

Figure 20: Coverage metric – (a) *C(MOGA, SPEA)*; (b) *C(SPEA, MOGA)* – sharing in criterion-space

In addition to criterion-space fitness sharing, a contemporary MOGA was also developed to incorporate solution-space sharing instead. Performance is empirically compared to that of the previous MOGA directly in Figure 21, and indirectly through comparison with the SPEA in Figure 22. Results are somewhat mixed. Poorer performance is observed on test functions 1, 2, 3, and 6 for the solution-space MOGA. Performance on ZDT-T4 and ZDT-T5 is evidently better, with a particularly marked difference for the multimodal test function.

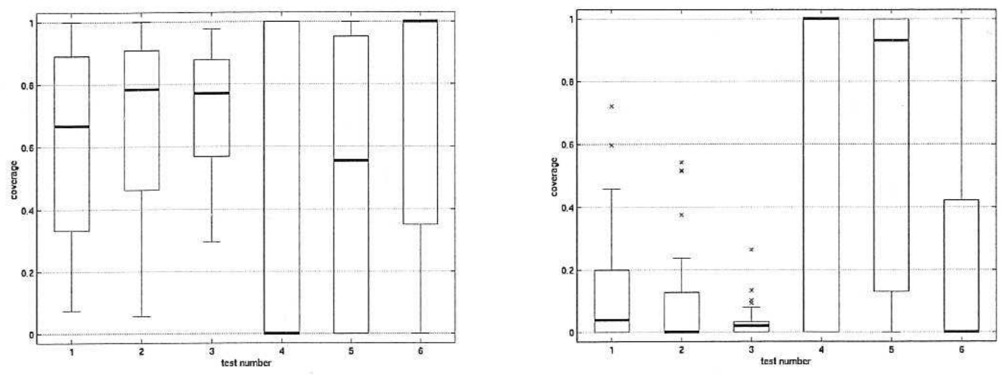

Figure 21: Coverage metric – (a) *C(objective, parameter)*; (b) *C(parameter, objective)*

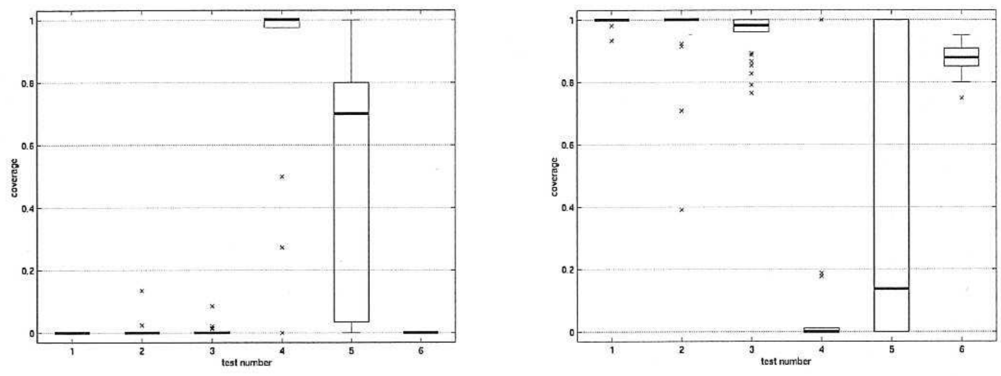

Figure 22: Coverage metric – (a) *C(MOGA, SPEA)*; (b) *C(SPEA, MOGA)* – sharing in solution-space

4.3 Variations

In the previous sub-section, the contemporary MOGA was found to produce significantly better results than the baseline MOGA across the test suite (apart from the deceptive ZDT-T5). However, since there are several differences between the two MOGA implementations, it is difficult to determine which settings, or combinations of settings, are acting as discriminants. Therefore, in this sub-section, attention in returned to the baseline MOGA and simple modifications are sought that lead to a significant improvement in performance.

Zitzler *et al* [2000] state that the performance of MOGA may improve using a lower mutation rate than that applied in their work (0.01), due to the fact that MOGA uses stochastic universal sampling as the selection mechanism whilst all other MOEAs use binary tournament selection with continuously updated sharing. The stated reason for this discrepancy was that MOGA requires a generational selection mechanism (indeed, the default setting *is* SUS), but it should be stressed that there is no inherent difficulty with implementing the latter selection mechanism within a generational framework. In these experiments, the accepted standard of an expected mutation of 1 bit per chromosome is used [Mühlenbein and Schlierkamp-Voosen, 1993]. For problems with thirty variables, each encoded over thirty bits, this results in a mutation rate of 1/900. This rate is a factor of 10 less than that used by Zitzler *et al*.

Preliminary investigations revealed that this adjustment in mutation rate did lead to improved performance, but that superior results were achieved by coupling this mutation rate with Gray encoding of decision variables (the coupling of these two factors produced better results than either individually). Note that choice of coding strategy is not applicable to ZDT-T5. To complete the MOGA variation, immigration of two random chromosomes was added. This did not produce a significant change in results, but random injection may guard against premature convergence. It is especially useful in time-varying optimisation problems, and is hence not regarded as a critical parameter for the set of test functions considered here.

Further preliminary investigations concerned with varying the niche size parameter for fitness sharing seemed to indicate that very low values of niche size produced the best results. Thus, as a further modification, fitness sharing was *removed* from the MOGA. In all, four separate MOGAs were implemented, as described in Table 6.

Name of MOGA	Description
bMOGA	The baseline MOGA, described in Section 3.2.
bMOGAxs	The baseline MOGA with niching removed.
mMOGA	The baseline MOGA with an expected mutation rate of 1 bit per chromosome, Gray encoding, and random injection of 2 individuals per generation.
mMOGAxs	mMOGA with niching removed.

Table 6: MOGA variations

Unified results for the first five runs of each algorithm, for each test problem, are shown in Figures 23 to 28. A purely visual analysis is used to determine a good-performing MOGA implementation. This MOGA is then compared to the SPEA using the coverage metric.

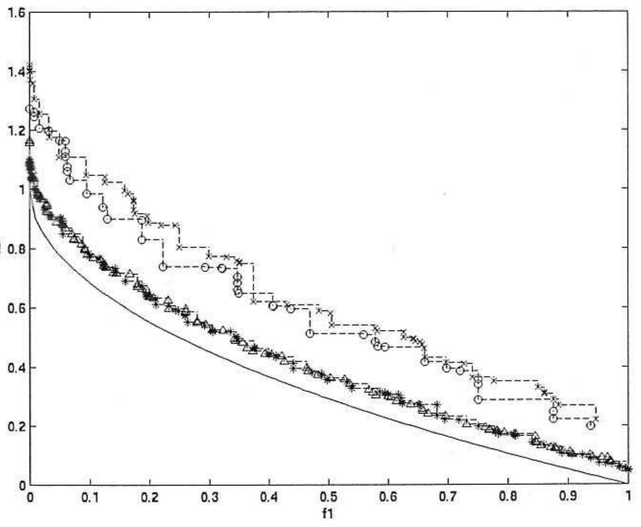

Figure 23: ZDT-T1 results (O – bMOGA, × – bMOGAxs, ✳ – mMOGA, △ - mMOGAxs)

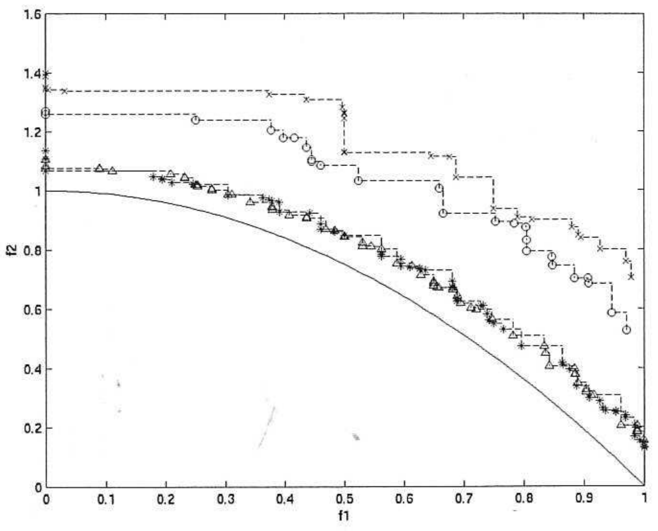

Figure 24: ZDT-T2 results (O – bMOGA, × – bMOGAxs, ✱ – mMOGA, △ - mMOGAxs)

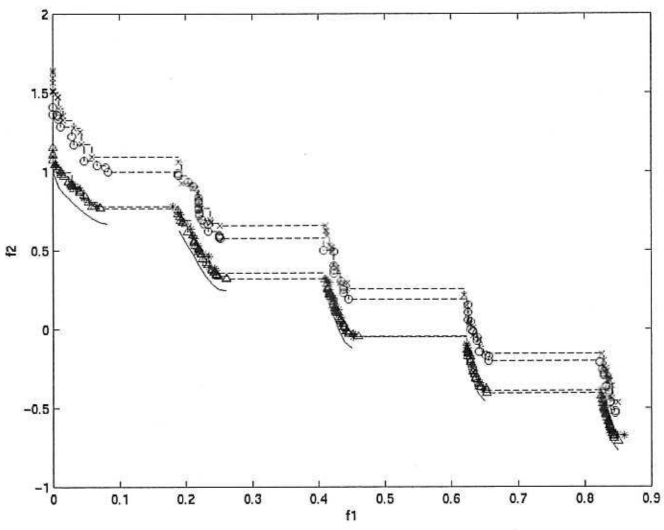

Figure 25: ZDT-T3 results (O – bMOGA, × – bMOGAxs, ✱ – mMOGA, △ - mMOGAxs)

The results obtained for the first three test functions suggest that adjustment of mutation rate and encoding choice to their widely recommended settings can lead to a noticeable improvement in results. Progress is largely in terms of accuracy, although some improvement in distribution can be seen. Note that the presence of criterion-space niching does not make a visually significant difference for these test functions. However, perhaps most surprisingly, MOGA without sharing can be regarded as the superior algorithm with respect to the coverage metric (coverage results for mMOGA and mMOGAxs are shown in Figure 29).

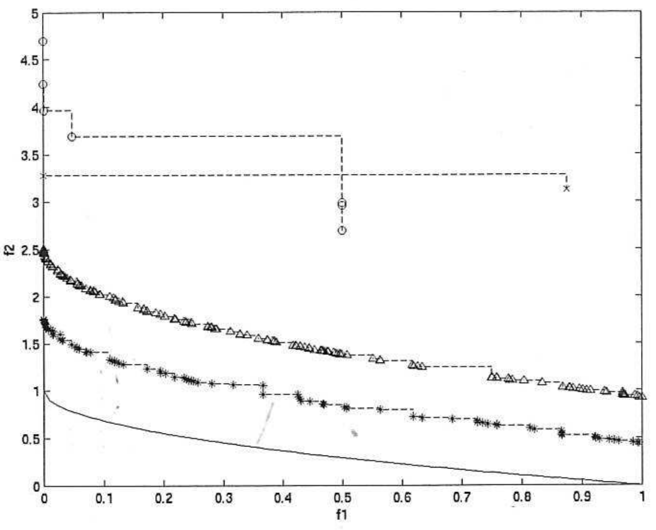

Figure 26: ZDT-T4 results (○ – bMOGA, × – bMOGAxs, ✳ – mMOGA, △ - mMOGAxs)

Results for ZDT-T4, as depicted in Figure 26, suggest that criterion-space niching *can* offer an improvement over a non-niching implementation, when combined with appropriate choices for mutation rate and encoding. This is also evident in the Figure 29 box plot, although some variation in coverage does exist.

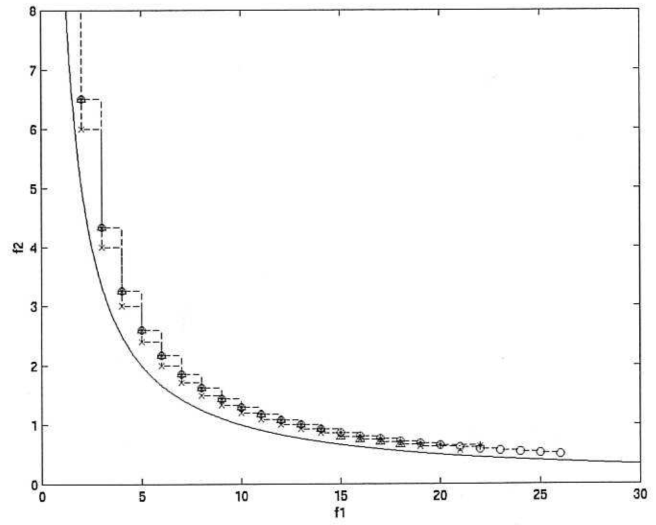

Figure 27: ZDT-T5 results (○ – bMOGA, × – bMOGAxs, ✳ – mMOGA, △ - mMOGAxs)

All four MOGAs exhibit very close levels of performance when applied to ZDT-T5, as shown in Figure 27. Recall that the baseline MOGA itself was found to perform well, when compared to other MOEAs, on this deceptive test function. In terms of accuracy, the best result obtained here is for bMOGA with niching removed, although good distribution is lost for larger values of objective f_1.

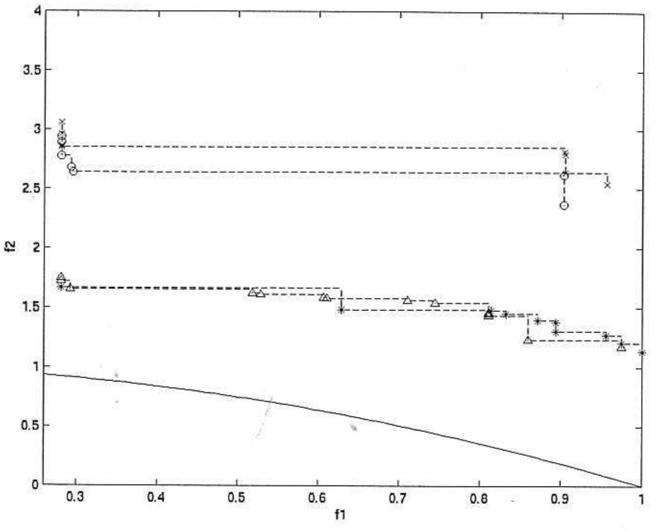

Figure 28: ZDT-T6 results (O – bMOGA, × – bMOGAxs, ✱ – mMOGA, △ - mMOGAxs)

Relative performance of the niching / non-niching variants would appear to be similar for ZDT-T6 as for the initial three test functions. However, in terms of coverage, mMOGA can be considered as the better performer (see the associated box plots in Figure 29).

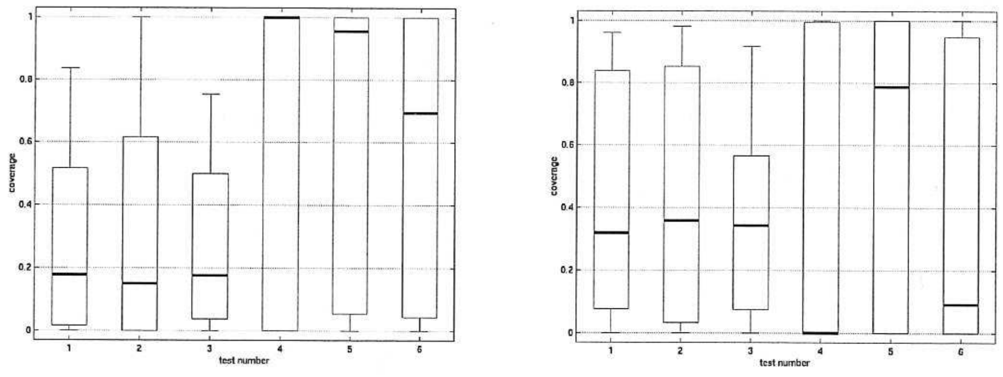

Figure 29: Coverage metric – (a) *C(mMOGA, mMOGAxs)*; (b) *C(mMOGAxs, mMOGA)*

It is apparent that the best results can be obtained by MOGAs with modified mutation and coding strategy, regardless of the application of a niching strategy. When implemented, criterion-space fitness sharing offers an improvement in results for the multimodal and non-uniform test functions. However, fitness sharing does prove somewhat detrimental to performance on the first three test functions. Since the magnitude of these downgrades is, arguably, more than offset by improvements on the other test problems, the mMOGA has been selected for comparison with the SPEA. The comparison is made, through use of the coverage metric, in Figure 30. SPEA still performs substantially better on the first three problems, although the completeness of its dominance has been reduced somewhat. On test function 4, MOGA is seen to produce a very high degree of coverage over SPEA. Performance on ZDT-T5 is still largely variable (this is a deceptive problem that generates mediocre results for all algorithms) although the median level of coverage of SPEA by MOGA is greater than the converse case. Finally, SPEA produces better results for the biased distribution problem.

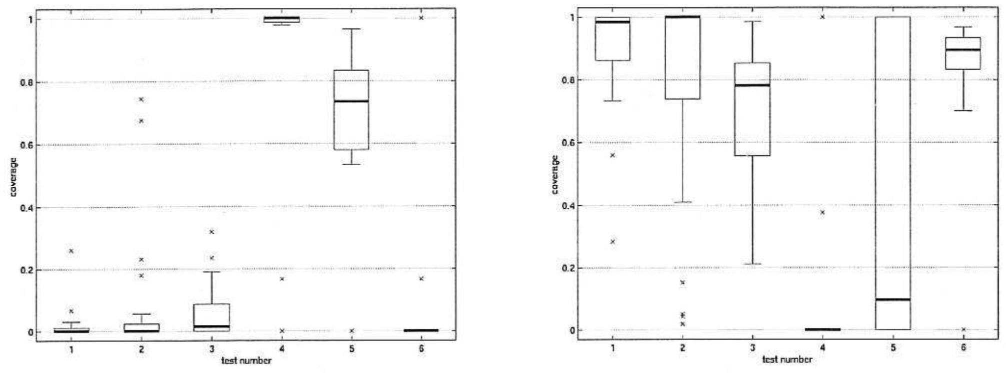

Figure 30: Coverage metric – (a) *C(mMOGA, SPEA)*; (b) *C(SPEA, mMOGA)*

The results presented in this section have implied that criterion-space niching (by means of Goldberg and Richardson's [1987] fitness sharing methodology) does not offer a definitive improvement in performance, in terms of either closeness to the true Pareto front or distribution along the trade-off surface. To conclude this part of the investigation, the performance of solution-space niching is established using an mMOGA. Solution-space niching has been implemented in the same manner as for the NSGA tested by Zitzler *et al* [2000]. Rank-wise Goldberg and Richardson [1987] fitness sharing was applied in the phenotypic solution-space (except for ZDT-T5, where a genotypic approach was required). Parameter α in Equation 3 was set equal to 2, whilst the niche size σ_{share} was set to 0.48862 (phenotypic) or 34 (genotypic), as per the specification provided by Zitzler *et al.*

Coverage results for the criterion-space and solution-space versions of mMOGA are shown in Figure 31. The latter MOGA is also compared to its non-sharing equivalent in Figure 32. The MOGA with solution-space niching is shown to produce significantly inferior results to both criterion-space mMOGA and the mMOGAxs across all test functions. However, its performance is still somewhat better than the baseline MOGA.

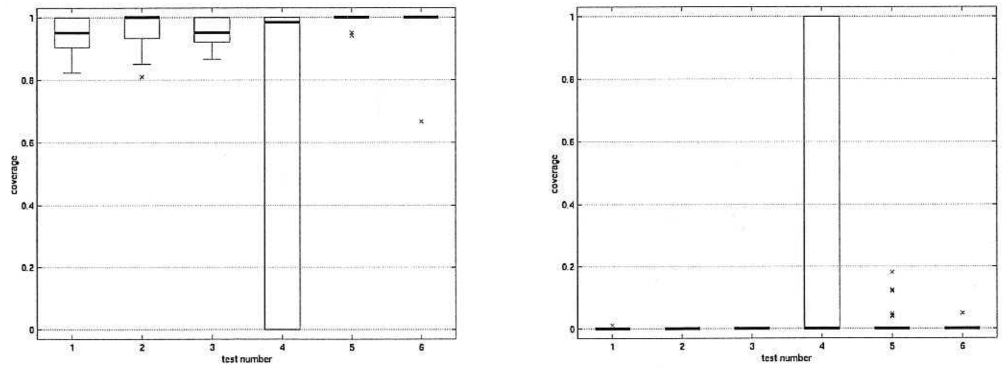

Figure 31: Coverage metric for mMOGA– (a) *C(criterion,solution)*; (b) *C(solution, criterion)*

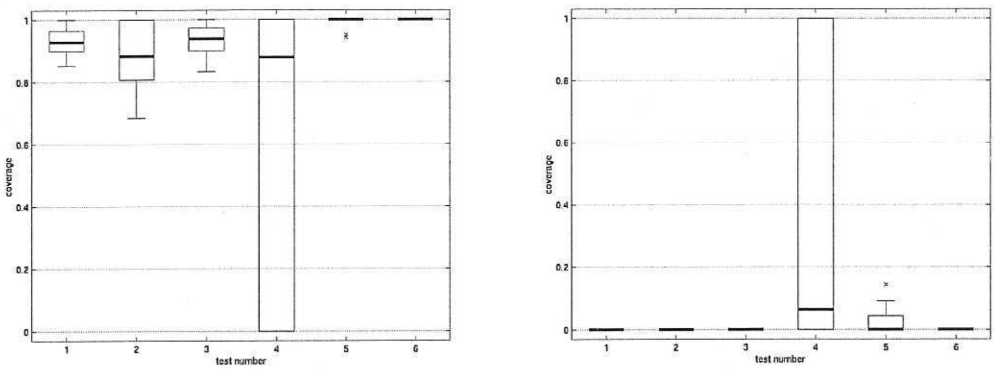

Figure 32: Coverage metric for mMOGA with solution-space fitness sharing –

(a) *C(mMOGAxs, mMOGA)*; (b) *C(mMOGA, mMOGAxs)*

4.4 Conclusion

In this section it was shown that by choosing a MOGA with settings widely regarded as standard by the research community, performance can be significantly improved across the set of test problems. However, of particular concern was the exhibited effect of fitness sharing. Whilst criterion-space niching offered some improvement in performance over several of the test functions, the use of niching was also found to be detrimental to performance on other test functions. Furthermore, the MOGA with solution-space fitness sharing exhibited very poor results, most worryingly even when compared to a non-niching algorithm. These are important results that receive further consideration in Section 6.

Note that SPEA still offers significant improvements over the MOGA, with the exceptions of ZDT-T4 and ZDT-T6. Further tuning of the MOGA settings could be attempted, but a more intelligent approach to problem solving is desirable. Five of the test functions in the suite consist of real-number variables. Hence, in the spirit of fitting the tool to the problem, it may be more appropriate to use a GA with real-number chromosomes. In Section 5, a real-coded MOGA is designed for the purposes of real-parameter function optimisation. In Section 6, the algorithm is tested and compared to the SPEA.

5 A real-coded genetic algorithm for function optimisation

5.1 Introduction

The modern approach to genetic algorithm design stresses the need for intelligent choice of objective function, representation, and search operators [Michalewicz and Fogel, 2000]. Choice of representation should be natural to the problem at hand. Many real-number decision variables, such as those in five of the ZDT test functions, can be intuitively represented by floating-point elements. In a classical genetic algorithm, these variables would be encoded as binary genotypes. Here, a conversion process is required to translate between the genotypic representation and phenotypic representation and *vice versa*. Floating-point representations do not require a conversion process, making them faster to manipulate. Furthermore, they permit greater precision than binary code, constrained only by machine precision.

In this section, a real-coded GA is devised for the purposes of real-parameter function optimisation. Particular attention is paid to the genetic operators that drive the search. In the following sub-section, a simple recombination operator from the literature is analysed. In the subsequent passage, a Gaussian-variant mutation operator is proposed. This adaptable operator is designed to work in synergy with the recombination operator and to effectively manage limits on decision variables.

5.2 Recombination

In this study, single-point crossover is used. This operator functions in exactly the same way for real-coded chromosomes as for the binary equivalent. Two chromosomes are chosen and a crossover point is randomly selected. Genetic material is exchanged at all loci to the right of the crossover point to produce two offspring. This process is illustrated in Figure 33.

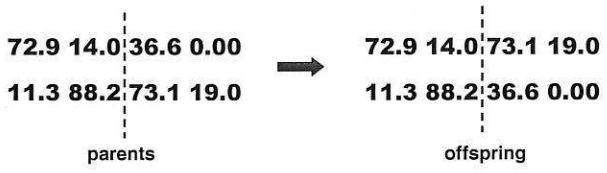

Figure 33: Single point crossover for real-number chromosomes

For real-coded chromosomes, the crossover point can only occur in-between decision variables (they are, essentially, atomic entities). For binary chromosomes, this restriction need not apply. It can be shown that, for real variables encoded as binary strings, if crossover occurs mid-parameter then the offspring will contain symmetrical perturbations of this parameter [Wright, 1991]. These new values for the decision variable cannot be achieved using standard point-wise crossover for real chromosomes. This is not necessarily a problem, but indicates that a high level of mutation may be required in order to improve genetic diversity. Without mutation, the values for the parameters that exist in the initial population would be all that are available. A similar problem does exist for binary representations, but tends to be less limiting (the values available depend on the chosen resolution and on the discrepancies between parents).

Note that the above behaviour does not apply to real-encoded GAs *per se*. Many real chromosome recombination operators have been developed in the literature, some of which seek to emulate the crossing-over of binary chromosomes. Herrera *et al* [1998] present a good review of potential approaches. They also provide a nice graphical summary of the rôle of crossover, reproduced in Figure 34. Crossover can act both as an exploratory operator and in its traditionally viewed rôle as an exploiter of genetic code. When applied to a binary encoding of parameters, the perturbations instilled by crossover can be viewed as a form of relaxed exploitation. Crossover operators have been devised for real-encodings of parameters that simulate this effect. These operators tend to be simpler to analyse than binary crossover, since the range covered by the perturbations of the latter approach is potentially non-continuous and depends on the variety, in terms of bits, between the parent chromosomes.

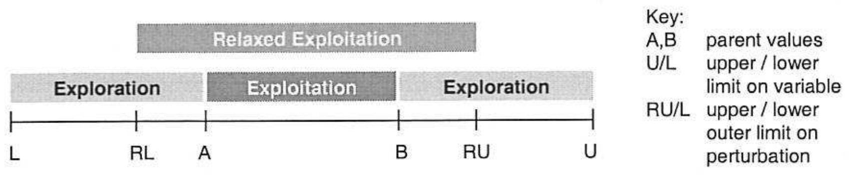

Figure 34: The rôle of crossover

5.3 Mutation

The mutation operator chosen in this study is similar to the standard element-wise operator for binary chromosomes. In the binary case, each element is probabilistically tested to undergo mutation. If mutation is to occur, the current value is swapped for an alternative. In the binary, or general discrete, case this is quite straightforward. However, in the real number case, an infinite set of alternatives exists. Gaussian mutation is an attractive method of choosing an alternative. This operator generates a new value based on a normal distribution, centred over the current value. The standard deviation defines the likelihood of generating a value close to the original. The advantage of this operator is that the standard deviation is amenable to variation during the search, hence providing an adaptable operator.

5.3.1 Maintaining feasibility with controlled bias

An evident problem with the Gaussian mutation operator is that it can produce infeasible solutions when applied to decision variable values towards the edge of the feasible region. This closeness to the edge can be defined in terms of standard deviations of the mutation operator. In this section, an approach to constraint handling is developed that ensures feasible solutions with minimal bias.

The simple solution to this problem is to crop any infeasible values to their nearest feasible equivalent. However, this will generate bias in the search towards the extreme values. In order to counter this, the standard deviation of the mutation function can be reduced, thus reducing the possibility of a required crop. Indeed, by predefining the number of standard deviations that must produce feasible results, it is possible to specify the acceptable level of bias.

Let n be the number of standard deviations (σ) from the current value that must produce feasible values, and let x be the shortest distance from a limit on the variable to the current value. If $n\sigma$ is greater than x, then σ must be rescaled to x/n in the direction of the limit (the value for σ away from the limit should remain unchanged). This procedure will create a discontinuity in the Gaussian function. Note that $n\sigma$ should be chosen so that a distance $n\sigma$ will be feasible in either direction from the central point between limits. Note that σ may vary over a chromosome, since the limits on decision variables may differ. Gaussian mutation applied to a single decision variable is illustrated in Figure 35.

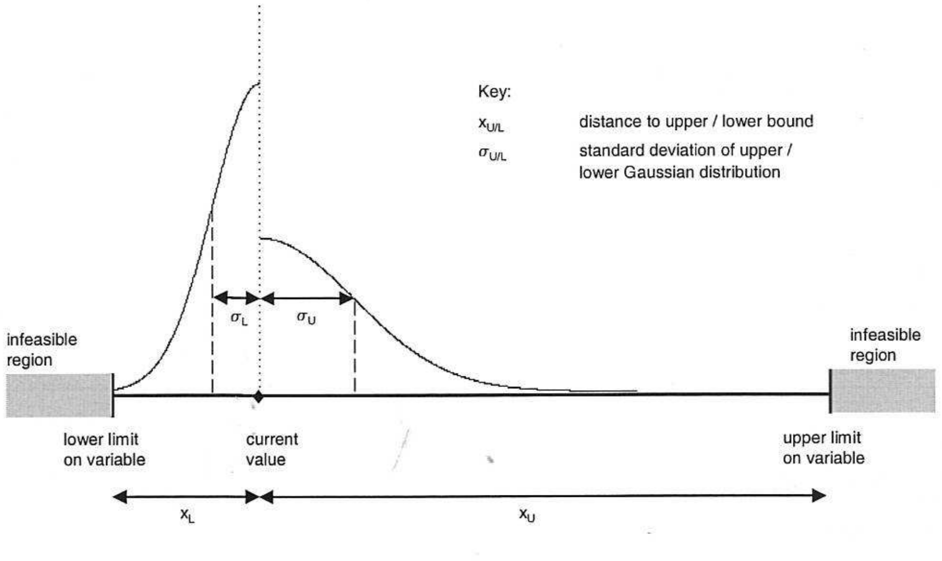

Figure 35: A Gaussian mutation operator

In this implementation, the probability of selecting either half of the distribution during mutation (defining the direction of mutation) varies with σ-rescaling. If no rescaling is necessary, this probability is 50-50. If rescaling is required, the probability of moving towards the nearest infeasible region is reduced to *(1/2)(x/nσ)*. This leads to more exploration away from the infeasible region when the infeasible region is close. In effect, the probability density under each half is rescaled in light of any necessary changes to σ.

Note that further action would be required for more complicated feasible regions on decision variables. This situation does not occur for the test functions under investigation here.

5.3.2 Adaptive mutation

In any search method, a trade-off exists between exploration of the search space and exploitation of promising regions of that space. The Gaussian mutation operator provides an opportunity to influence this trade-off by adapting the standard deviation over the course of a run. In this study, a sigmoidal function (Equation 8) is used to vary the standard deviation over the generations. The parameter a in Equation 8 defines the slope of the *S*-shape of the sigmoid. This function, coupled with a suitably large σ (for example, 40% of the search space for each variable) provides copious global exploration early in the search and much local exploration late in the search. The exact levels are defined by the a parameter. Larger values of a will detain the search at a global level and micro level for longer (at the expense of intermediate levels of mutation).

$$s(gen) = \frac{e^{-a\left(\frac{gen}{g_{max}}-\frac{1}{2}\right)}}{1+e^{-a\left(\frac{gen}{g_{max}}-\frac{1}{2}\right)}} \tag{8}$$

where s is the mutation scaling factor,

gen is the current generation number,

g_{max} is the maximum generation number, and

a is a shaping parameter.

5.4 Incorporation into a genetic algorithm

The genetic representation and operators described above must be integrated with selection, fitness assignment, and reinsertion mechanisms in order to construct a genetic algorithm. In the multiobjective case, Fonseca and Fleming's [1993] Pareto ranking was applied, prior to linear fitness assignment.

In this work, two selection mechanisms were considered: stochastic universal sampling (SUS) [Baker, 1987] and deterministic BGA selection [Mühlenbein and Schlierkamp-Voosen, 1993]. The BGA methodology of selecting the best individuals for breeding works well in the single objective case, but was not so suitable for multiobjective problems where a distribution of optimal (in the Pareto sense) solutions was to be identified. SUS was found to be better for this latter case.

When considering reinsertion, elitism was again more suitable for single objective problems than multiobjective tasks, where large numbers of nondominated individuals can lead to a directionless search. In this implementation, all offspring replaced all parents. Two randomly generated chromosomes were introduced at each generation to maintain a constant supply of diversity.

5.5 Conclusion

In this section, a genetic algorithm for the solution of real-parameter function optimisation problems was designed from basic evolutionary computing principles. In particular, the GA uses a natural expression for potential solutions, and offers customised search operators to support this representation. The Gaussian mutation operator developed during this study permits an explicit and adaptive specification of the search neighbourhood, thus providing a degree of control over the exploration-exploitation trade-off. In the following section, the GA is integrated into Fonseca and Fleming's [1993] MOGA framework, and is subsequently applied to the solution of the real-parameter ZDT problems.

6 Real-encoded MOGA results and analysis

6.1 Introduction

The real-coded MOGA, developed in the previous section using the guidelines suggested by Michalewicz and Fogel [2000], is now applied to the five real-parameter problems in the Zitzler *et al* [2000] test suite. The settings for the MOGA are detailed in Table 7. Two implementations were considered, namely: with and without criterion-space niching. These MOGAs are labelled *rMOGA* and *rMOGAxs* respectively.

MOEA parameter	Setting
General GA	
Population size	100
Total generations	250
Coding	Real number encoding (accuracy bounded by machine precision)
Selection	Stochastic universal sampling [Baker, 1987]
Recombination	Single-point binary crossover, probability = 0.8 (crossover points are restricted to between decision variables)
Mutation	Gaussian mutation with initial standard deviation of 40% of variable range. The standard deviation is adapted using sigmoidal scaling (as a function of %age generations complete). Sigmodal parameter = 17.5. Bias: at least one standard deviation from the current value is required to be within the limits of the decision variable (by appropriate modification if necessary). Expected mutation rate of 1 element per chromosome.
Generational gap	Zero
Random injection	2 random chromosomes per generation
Elitism	None
Multiobjective GA	
Fitness assignment	Fonseca and Fleming's [1993] multiobjective ranking (see Section 2.2). Transformation from rank to fitness using linear fitness assignment with rank-wise averaging.
External population	Off-line storage of nondominated solutions
Niching	
Fitness sharing	When fitness sharing is implemented, Goldberg and Richardson's [1987] method is used, with niche size calculated using the equation in [Deb and Goldberg, 1989].
Mating restriction	None.

Table 7: Real-coded MOGA settings

Niching is implemented according to the format given in [Goldberg and Richardson, 1987]. A triangular fitness sharing function (see Equation 3) has been selected ($\alpha = 1$) and ten niches are aimed for ($q = 10$). The niche size has been calculated using the formula devised by Deb and Goldberg [1989], thus:

$$\sigma_{share} = \frac{\sqrt{\sum_{k=1}^{p}\left(x_{k,max} - x_{k,min}\right)^2}}{2\sqrt[p]{q}} = \frac{\sqrt{\sum_{k=1}^{2}(1-0)^2}}{2\sqrt[2]{10}} = 0.2236$$

The criterion-space results are normalised to $[0\ 1]^2$ space prior to application of fitness sharing. This normalisation is achieved by utilising the population maximum and minimum values for each objective at each generation. Hence, the transformation to normalised space is a dynamic operator.

Note that fitness sharing is applied in criterion-space rather than solution-space in order to obtain a good distribution of solutions across the approximated Pareto front. This will maximise the amount of information concerning the trade-off between the two objectives to be presented to the decision-maker. In this set of tests, performance is not directly analysed in terms of distribution in solution-space. Hence, the quality of this distribution is not of immediate concern.

6.2 Results

The results for the two real-coded MOGA implementations are shown in the subsequent figures. Note that test function ZDT-5 is not included in this study since it does not involve real variables. Results for the SPEA, as obtained by Zitzler *et al* [2000], are displayed for reference purposes. The Pareto optimal unified results of the first five runs of each algorithm are shown for each test function in Figures 36 through 40. Box plots of the thirty run-wise coverage metrics are shown for each test function in Figures 41 to 43. In Figure 41, the twin coverage results for rMOGA and rMOGAxs are shown. In Figure 42, rMOGA is compared to SPEA, whilst in Figure 43 rMOGAxs is compared to SPEA.

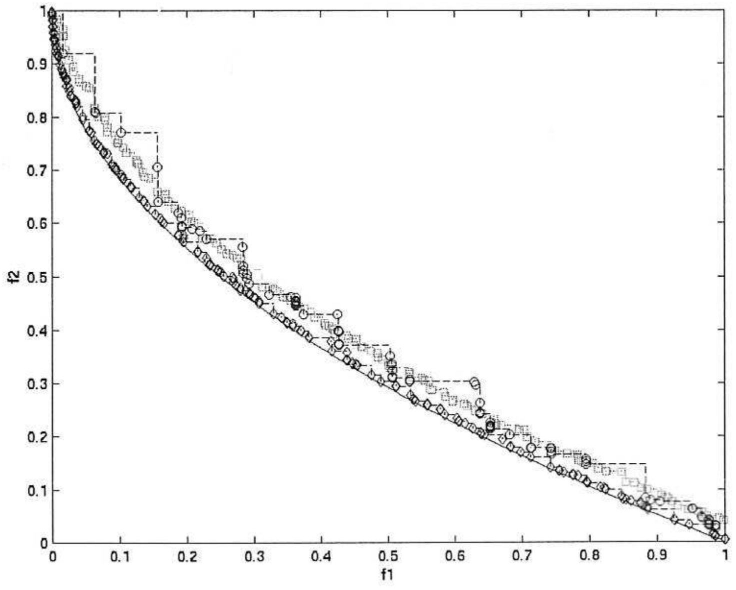

Figure 36: ZDT-T1 results: ○ - rMOGA; ◇ - rMOGAxs; □ - SPEA

As expected, approximately ten sub-populations are visible for rMOGA on ZDT-T1, as shown in Figure 36. In terms of accuracy, rMOGA is somewhat on a par with SPEA.

However, it loses out in terms of smoothness of distribution. This is perhaps expected since fitness sharing was set to provide ten niches across the Pareto front. rMOGAxs exhibits superior performance to SPEA in terms of both accuracy and distribution, as confirmed by the coverage results in Figure 43. The smoothness of the distribution can be attributed to the fact that niching was not implemented for this MOGA. Note that, in an MOEA that utilises fitness sharing, smoothness would be expected to increase as the niche size decreases. Results for the rMOGAxs are slightly concentrated towards $f_1 \rightarrow 0$. This may be due to the lack of penalties for overcrowding. However, total convergence to one part of the front has been avoided without the need for explicit action.

rMOGAxs may be producing more accurate results than rMOGA because it has an increased level of selective pressure. The fitness sharing algorithm in rMOGA tends to reduce the average fitness of the population. Since an individual's fitness relates directly to the expected number of times it is selected for reproduction, reducing fitness reduces the selective pressure. Hence, after a fixed number of function evaluations, rMOGAxs would be expected to have converged more than rMOGA. Thus, rMOGAxs could be postulated to be closer to the front but, given further generations, rMOGA would produce this level of accuracy.

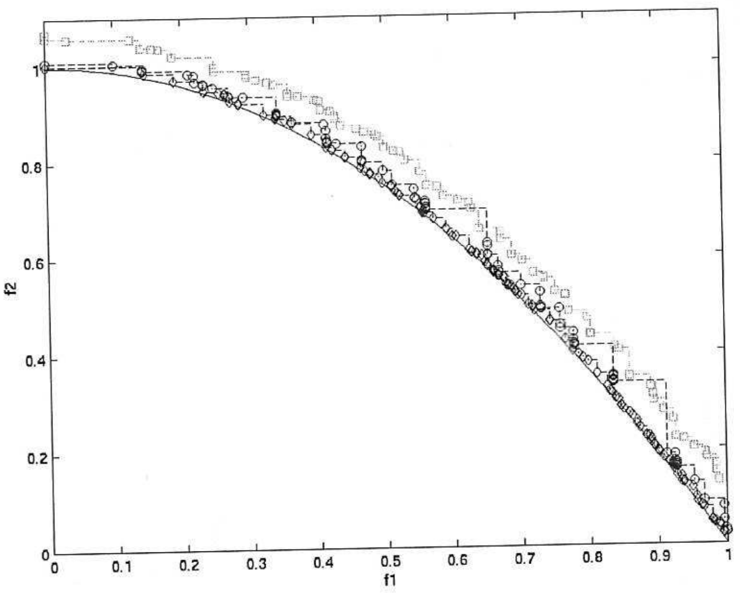

Figure 37: ZDT-T2 results: ○ - rMOGA; ◇ - rMOGAxs; □ - SPEA

Both real-coded MOGAs struggle to produce a uniform distribution of solutions as $f_1 \rightarrow 1$ for ZDT-T2 (see Figure 37). This behaviour is also exhibited by all MOEAs in Zitzler *et al*'s [2000] study. rMOGAxs comprehensively outperforms SPEA, whilst rMOGA is relatively on a par with the latter algorithm. These results are evident both from Figure 37, and the coverage results in Figure 42 and Figure 43.

Results for ZDT-T3 (refer to Figure 38) are very tightly packed. However it can be seen that rMOGAxs outperforms SPEA in terms of closeness to the global front. This result is borne out by the associated box plots in Figure 43.

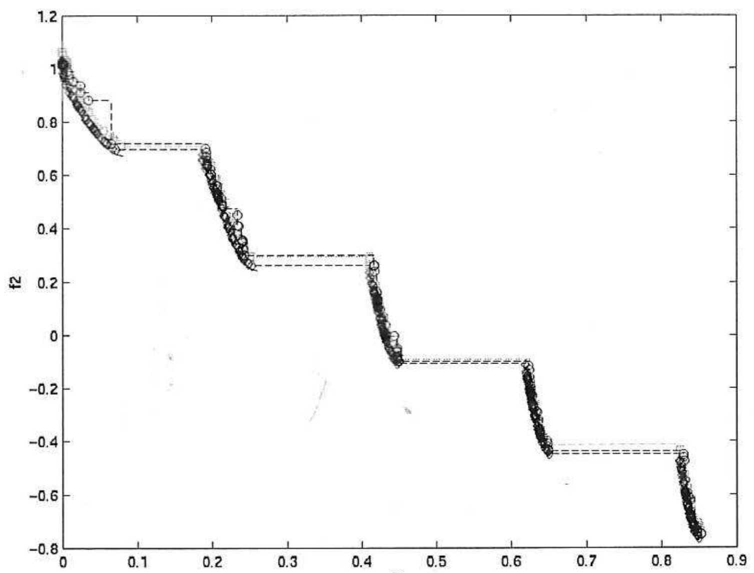

Figure 38: ZDT-T3 results: O - rMOGA; ◇ - rMOGAxs; □ - SPEA

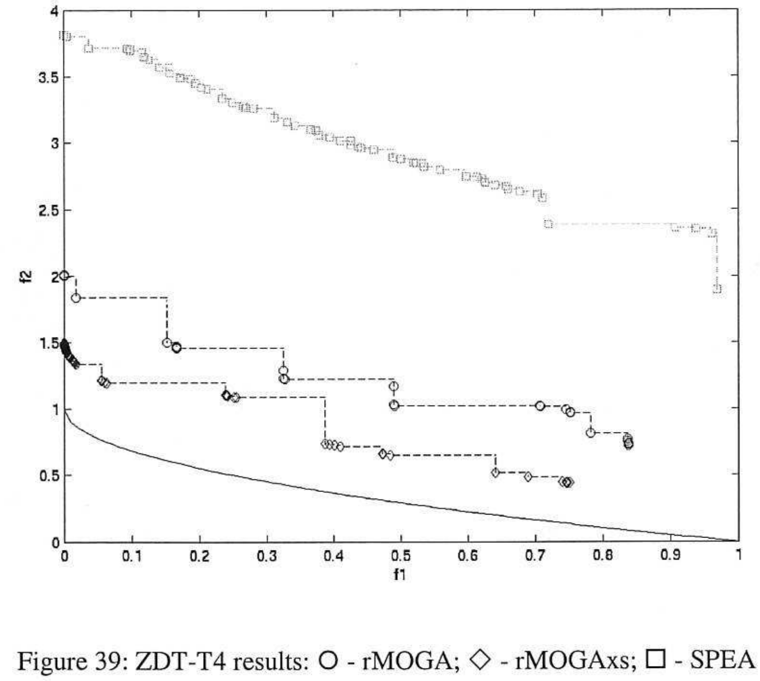

Figure 39: ZDT-T4 results: O - rMOGA; ◇ - rMOGAxs; □ - SPEA

Results for ZDT-T4 are shown in Figure 39. Both real-coded MOGAs exhibit markedly better performance than the SPEA on this test problem. The coverage box plots in Figure 42 and Figure 43 are a testament to this. Niches can apparently be seen for rMOGA although, somewhat curiously, sub-populations are also present for the non-niching rMOGAxs. Note that all MOEAs have difficulty in representing the trade-off surface as $f_1 \rightarrow 1$, although SPEA has identified the extreme values.

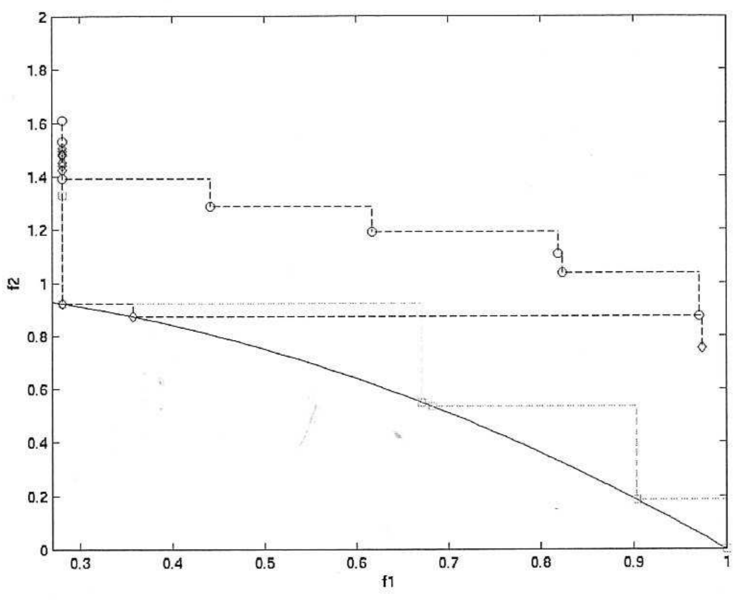

Figure 40: ZDT-T6 results: ○ - rMOGA; ◇ - rMOGAxs; □ - SPEA

SPEA identifies the global front for ZDT-T6 with very high accuracy and a uniformly spread distribution, albeit with significant gaps between solutions. Neither MOGA is able to match SPEA for repeatable accuracy, although rMOGAxs achieves similar results in the densest area of criterion space ($f_1 \rightarrow 0$). rMOGA finds a good distribution of solutions, demonstrating the key benefit of fitness sharing on biased problems, but with comparatively low accuracy (possibly for the reasons outlined earlier).

Note that, despite its apparent domination regarding the unified results of the first five runs on ZDT-T6, SPEA does not comprehensively outperform rMOGAxs across all thirty replications. Indeed the median value for coverage of SPEA by rMOGAxs is actually greater than the converse case (see Figure 43).

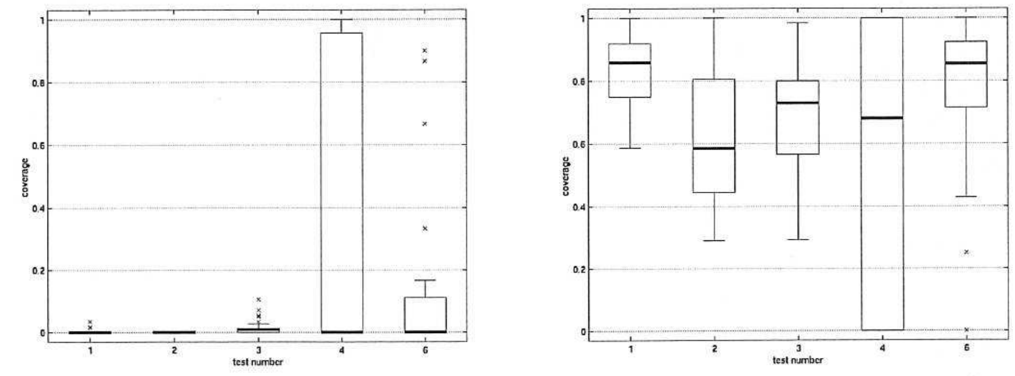

Figure 41: Coverage metric – (a) *C(rMOGA, rMOGAxs)*; (b) *C(rMOGAxs, rMOGA)*

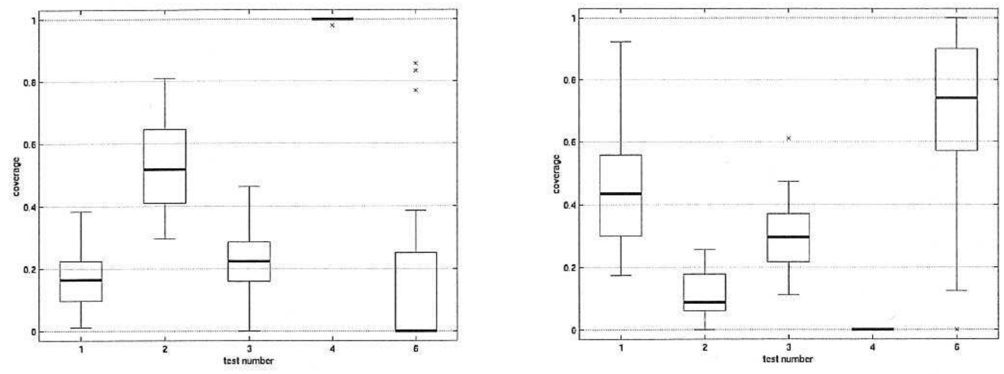

Figure 42: Coverage metric – (a) *C(rMOGA, SPEA)*; (b) *C(SPEA, rMOGA)*

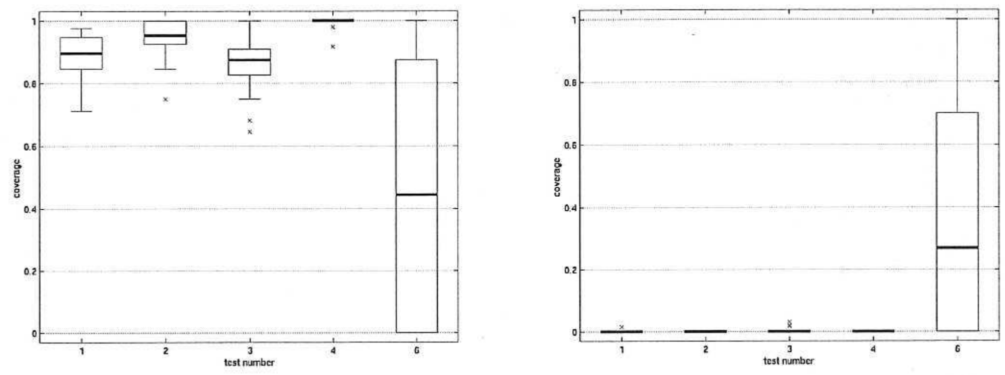

Figure 43: Coverage metric – (a) *C(rMOGAxs, SPEA)*; (b) *C(SPEA, rMOGAxs)*

6.3 Conclusion

In conclusion, the real-encoded MOGA developed for the purposes of real-parameter function optimisation performs better overall than any of the MOEAs examined by Zitzler *et al* [2000]. This does not mean that the MOGA, as a technique, is superior to other MOEAs. Real-coded versions of other MOEAs are likely to produce similar results. The critical conclusion to be drawn, as elucidated by Michalewicz and Fogel [2000], is that an evolutionary computing approach to problem-solving requires care and thought in order to achieve good results, given that the problem is considered amenable to an evolutionary solution. Mass empirical testing of algorithms, with lumped settings, is unlikely to provide a confident basis from which a hierarchy of MOEAs can be deduced.

Results in this section have confirmed, as indicated by results from previous sections, that the application of niching via fitness sharing is not producing the benefits anticipated of such an approach. This is discussed further in the following, and concluding, section of this report.

7 Conclusion

7.1 General conclusions

In this report, the multiobjective genetic algorithm (MOGA) was analysed using the test suite proposed by Zitzler *et al* [2000]. The emulation of Zitzler *et al*'s *FFGA* MOGA implementation, labelled here as the *bMOGA*, was found to outperform the results published in that article. This discrepancy may be attributed to differing interpretations of the paper by Fonseca and Fleming [1993] upon which the algorithm is based. However, bMOGA was clearly uncompetitive with other popular MOEAs, in particular the Strength Pareto Evolutionary Algorithm (SPEA) developed by Zitzler and Thiele [1999].

The specification of the bMOGA does not match that of contemporary MOGA implementations. When a standard MOGA was applied to the test problems, significant increases in performance were evident although, as before, performance did not compare favourably to that of the SPEA on the majority of the test problems.

It has been stressed throughout that MOGA represents a methodology rather than a specific, unalterable, algorithm. In light of this, and using the guidelines described by Michalewicz and Fogel [2000], a MOGA has been designed specifically for real-parameter multiobjective function optimisation. This MOGA has been shown to outperform all the MOEAs assessed by Zitzler *et al* [2000]. This is not a statement to the effect that MOGA is the best MOEA. All that has been demonstrated is that, through thoughtful use of the evolutionary computing framework, an MOEA can be designed in a straightforward fashion to provide good solutions to real-parameter bi-objective test problems. The latest generation of algorithms, including Deb *et al*'s [2000] NSGA II, Zitzler *et al*'s [2001] SPEA II, and the Pareto Archived Evolution Strategy (PESA) developed by Knowles and Corne [1999], are all expected to perform well on this set of benchmark problems.

This is not to say that the development of test problems and the analysis of performance is not a worthwhile activity. Indeed, it is an essential part of the collective research effort. It is important to identify what action is required to effectively manage various problem attributes, such as multimodality and non-uniformity. However, it is equally important to devise methods for *identifying* these attributes in the cost landscapes of real-world problems. Test problems are required that are tractable yet sufficiently complex to offer realistic multiobjective scenarios. In particular, these problems must feature more than a small number of objectives. The recent work by Deb *et al* [2001] represents an important step in this direction.

7.2 Niching and the ZDT test problems

The results presented in this report suggest that several of the ZDT test problems do not require niching, and furthermore that niching can degrade performance given a limited number of function evaluations. However, on problems where a random scattering of solutions produces a biased distribution on the front, criterion-space niching has a positive impact. Criterion-space niching was also found to be useful for problems with a multimodal system of fronts. Niching will also be valuable when the Pareto front is relatively large with respect to the total search environment and the MOEA population size. In these circumstances, the MOEA can be swamped by a high number of equal-fitness solutions (essentially reducing the algorithm to the status of a random walk and thus increasing the likelihood of genetic drift). Real-world applications show that the possibility of this occurrence increases with the number of objectives [Fonseca and Fleming, 1998], perhaps suggesting why it has not been evident for these two-objective test functions. Niching is one of the tools to counter this problem, along with preference articulation [Fonseca and Fleming, 1998] and structured-population approaches.

Horn *et al* [1994] communicate the sensible belief that sharing should be performed in the space "we care more about". Hence, if a good distribution in criterion-space is required, then criterion-space niching should be applied. It seems somewhat unusual that the NSGA in Zitzler *et al* [2000] should use solution-space niching when performance was clearly defined in the objective domain. Also, if niching is functioning as expected, sub-populations should surely be evident in the results. This was true for the MOGAs with criterion-space niching investigated in this report. The empirical results suggest that SPEA does not produce niches in the Goldberg and Richardson [1987] sense. Indeed its performance more closely resembles that of an MOEA without a niching scheme. This may be of concern for application to larger test problems, especially where none of the objectives are completely defined by a single decision variable. However, it may also be true that the SPEA's non-conventional method of fitness assignment *is* actively producing a good distribution in criterion-space. If so, the SPEA would appear to offer the further benefit of a continuous distribution, rather than the somewhat forced sub-population structure that results from fitness sharing.

Note that sub-populations are visible in the results for the NSGA presented in Zitzler *et al* [2000], despite the fact that NSGA was implemented with solution-space niching. This result may be an artefact of the test problems. Since one of the objectives is actually a decision variable, a direct correlation exists between distances in solution-space and distances in criterion-space. Thus, sharing in solution-space is closely related to sharing in criterion-space. On problems with more complicated functional relationships between the two domains, niching in solution-space could not be expected to produce a good distribution of sub-populations in criterion-space. Note that solution-space sharing is still important in an MOEA for its original purpose: avoiding premature convergence within the cost landscape. Furthermore, as highlighted by Deb [1999], solution-based sharing should result in a diverse set of solutions to present to the decision-maker.

The remarkable performance of algorithms that do not use fitness sharing should be regarded with caution. These results may be an artefact of the test problems. Since one decision variable has complete control over the distribution of one objective, genetic drift towards one part of the Pareto front is not a great problem. This is especially true for real-coded chromosomes because a single mutation can potentially cover the entire range of the objective (whereas for a binary chromosome, the range would depend on the significance of the mutated bit). In the test functions proposed by Deb [1999], once good performance is achieved in f_2, a single decision variable can be used to manipulate f_1 to generate solutions across the Pareto front (although decision variable mapping can be introduced to prevent this, at a greater cost in terms of test function complexity). However in situations where good performance, in terms of accuracy and distribution, must be steadily developed using a building-block approach, premature convergence in an objective can pose a severe problem. It would be unlikely that random mutation would prove sufficient to refresh the population. Niching will prove very valuable in these latter cases.

Acknowledgements

The authors would like to express their gratitude to Eckart Zitzler for the on-line availability of the performance data from the Zitzler *et al* [2000] study. They would also like to thank Dr Zitzler for his response to queries on the associated paper.

Data

The MOGA results presented in this report are available for download from the following site: http://www.shef.ac.uk/~acse/research/students/r.c.purshouse/.

References

Baker, J. E., 1987, *Reducing bias and inefficiency in the selection algorithm*, Proceedings of the Second International Conference on Genetic Algorithms, Ed. Grefenstette, J. J., Lawrence Erlbaum Associates, pp14-21.

Chipperfield, A. and Fleming P., 1996, *Multiobjective Gas Turbine Engine Controller Design Using Genetic Algorithms,* IEEE Transactions on Industrial Electronics, Vol. 43, No. 5, pp1-5, October 1996.

Cleveland, W. S., 1993, *Visualizing Data*, Hobart Press, Summit, New Jersey.

Coello Coello, C. A., 1999, *A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques*, Knowledge and Information Systems. An International Journal, Vol. 1, No. 3, pp269-308, August 1999.

Dakev, N. V., Whidborne, J. F., Chipperfield, A. J., and Fleming, P. J., 1997, *Evolutionary H-infinity design of an electromagnetic suspension control system for a maglev vehicle*, Proceedings of the Institution of Mechanical Engineers, Vol. 211, Part I, pp345-355.

Deb, K., 1999, *Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems*, Evolutionary Computation, Vol. 7, No. 3, pp205-230.

Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T., 2000, *A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II*, KanGAL report 200001, Indian Institute of Technology, Kanpur, India.

Deb, K. and Goldberg, D. E., 1989, *An Investigation of Niche and Species Formation in Genetic Function Optimization*, in: Schaffer, J. D. (Ed.), Proceedings of the Third International Conference on Genetic Algorithms, pp42-50, Morgan Kaufmann, San Francisco, CA.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E., 2001, *Scalable Test Problems for Evolutionary Multi-Objective Optimization*, TIK-Technical Report No. 112, ETH Zürich, Switzerland.

Fonseca, C. M. and Fleming, P. J., 1993, *Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization*, Genetic Algorithms: Proceedings of the Fifth International Conference, Morgan Kaufmann, San Mateo, CA, pp416-423.

Fonseca, C. M. and Fleming, P. J., 1995, *Multiobjective Genetic Algorithms Made Easy: Selection, Sharing and Mating Restriction*, Proceedings of GALESIA '95 (Genetic Algorithms in Engineering Systems: Innovations and Applications), pp45-52.

Fonseca, C. M. and Fleming, P. J., 1998, *Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms – Part I: A Unified Formulation* and *Part II: Application Example*, IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, Vol. 28, No. 1, pp26-37 and pp38-47, January 1998.

Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Reissue, Addison-Wesley Publishing Company.

Goldberg, D. E. and Richardson, J., 1987, *Genetic Algorithms with Sharing for Multimodal Function Optimization*, in: Grefenstette (Ed.), Proceedings of the Second International Conference on Genetic Algorithms, pp41-49.

Griffin, I. A., Schroder, P., Chipperfield, A. J., and Fleming, P. J., 2000, *Multi-objective optimization approach to the ALSTOM gasifier problem*, Proceedings of the Institution of Mechanical Engineers, Vol. 214, Part I, pp453-468.

Haas, O. C. L., Burnham, K. J., and Mills, J. A., 1997, *On improving physical selectivity in the treatment of cancer: a systems modelling and optimisation approach*, Control Engineering Practice, Vol. 5, No. 12, pp 1739-1745.

Herrera, F., Lozano, M., and Verdegay, J. L., 1998, *Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis*, Artificial Intelligence Review, Vol. 12, pp265-319.

Horn, J. and Nafpliotis, N., 1993, *Multiobjective Optimization Using the Niched Pareto Genetic Algorithm*, Technical Report 930005, Illinois Genetic Algorithms Laboratory (IlliGAL), University of Illinois, Urbana, Illinois.

Horn, J., Nafpliotis, N., and Goldberg, D. E., 1994, *A Niched Pareto Genetic Algorithm for Multiobjective Optimization*, in Michalewicz, Z. (Ed.), Proceedings of the First IEEE Conference on Evolutionary Computation, pp82-87, IEEE Press, Piscataway, NJ.

Knowles, J. D. and Corne, D. W., 1999, *The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimisation*, in 1999 Congress on Evolutionary Computation, IEEE Service Center, pp98-105.

Michalewicz, Z. and Fogel, D. B., 2000, *How to Solve It: Modern Heuristics*, Corrected Second Printing, Springer-Verlag, Berlin.

Mühlenbein, H. and Schlierkamp-Voosen, D., 1993, *Predictive Models for the Breeder Genetic Algorithm*, Evolutionary Computation, Vol. 1, No. 1, pp25-49.

Obayashi, S., Sasaki, D., Takeguchi, Y., Hirose, N., 2000, *Multiobjective Evolutionary Computation for Supersonic Wing-Shape Optimization*, IEEE Transactions on Evolutionary Computation, Vol. 4, No. 2, July 2000, pp182-187.

Rodríguez-Vázguez, K., Fonseca, C. M., and Fleming P. J., 1997, *Multiobjective Genetic Programming: A Nonlinear System Identification Application*, Late Breaking Papers at the 1997 Genetic Programming Conference, pp207-212.

Schaffer, J. D., 1984, *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*, PhD thesis, Vanderbilt University.

Schroder, P., Green, B., Grum, N., and Fleming, P. J., 2001, *On-line evolution of robust control systems: an industrial active magnetic bearing application*, Control Engineering Practice, Vol. 9, No. 1, pp37-49, January 2001.

Silverman, B. W., 1986, *Density Estimation for Statistics and Data Analysis*, Monographs on Statistics and Applied Probability 26, Chapman and Hall.

Srinivas, N. and Deb, K., 1994, *Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms*, Evolutionary Computation, Vol. 2, No. 3, pp221-248.

Van Veldhuizen, D. A. and Lamont, G. B., 2000, *Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art*, Evolutionary Computation, Vol. 8, No. 2, pp125-147.

Wright, A. H., 1991, *Genetic algorithms for real parameter optimisation*, in: Rawlins, G. J. E. (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, pp205-218.

Zitzler, E., Deb, K., and Thiele, L., 2000, *Comparison of Multiobjective Evolutionary Algorithms: Empirical Results*, Evolutionary Computation, Vol. 8, No. 2, pp173-195.

Zitzler, E., Laumanns, M., and Thiele, L., 2001, *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*, TIK-Technical Report No. 103, ETH Zürich, Switzerland.

Zitzler, E. and Thiele, L., 1999, *Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach*, IEEE Transactions on Evolutionary Computation, Vol. 3, No. 4, pp257-271.