Multiobjective Genetic Algorithms Made Easy: Selection, Sharing and Mating Restriction

Carlos M. Fonseca and Peter J. Fleming

University of Sheffield, UK

1 Introduction

The populations of current approximations maintained by Genetic Algorithms (GAs) and other evolutionary approaches confer the ability to concurrently search for multiple solutions to given problems. This is particularly relevant in engineering, where multiple and often conflicting objectives seldom define optimal solutions uniquely. However, this ability is overlooked in most current applications of GAs in engineering, and GAs are used simply for their generality and robustness as an alternative to, but in the same spirit of, more restrictive conventional optimizers. Different objectives are thus analytically combined into a single function prior to optimization, and the GA applied.

This paper aims to illustrate how an existing GA can be modified and set up to explore the relevant trade-offs between multiple objectives with a minimum of effort. While Pareto and Pareto-like ranking schemes [1, 2] can be easily implemented, current guidelines on the associated set-up of techniques such as sharing and mating restriction [3, 2] are intricate and/or based on more or less rough assumptions about the cost landscape, making them impractical. However, if fitness sharing is reinterpreted as a technique involving the estimation of the population density at the points defined by each individual by so-called *kernel* methods [4], the setting of the sharing parameter comes to depend only on the size and current distribution of the population, and not on the problem. Kernel density estimation [4], a technique from statistics and data analysis, will be introduced and shown to find direct application in sharing and mating restriction, simplifying implementation and avoiding the introduction of any more tunable parameters in the GA formulation.

After a brief introduction to multiobjective optimization and a discussion of preference articulation in GAs, the main differences between single-objective and multiobjective GAs are highlighted, and the conversion of an existing GA into a multiobjective GA described by means of an example. Simple experimental results are presented towards the end of the paper.

2 Multiobjective optimization and preference articulation

Most engineering problems are characterized by several non-commensurable and often competing objectives to be optimized. Due to the trade-offs involved, such problems usually have no unique, perfect solution. Instead, they admit a set of equally valid, or non-dominated, alternative solutions, which is known as the Pareto-optimal set [5]. These solutions are such that improvement in any objective can only be achieved at the expense of degradation in other objectives, and can only be discriminated on the basis of expert knowledge of the problem. This may include the understanding of the importance of certain objectives relative to others or the need to meet given specifications, for example.

Although non-dominated solutions can generally be obtained through optimization, expressing informal design preferences in terms of a sufficiently well behaved cost function, as expected by many conventional optimizers, is not always easy. In particular, unimodality requirements imply that all decisions must be made prior to optimization. If the solution produced by the optimizer is not satisfactory, the cost function must be changed and the process repeated.

Genetic Algorithms in Engineering Systems: Innovations and Applications
12-14 September 1995, Conference Publication No. 414, © IEE, 1995

On the other hand, Genetic Algorithms only require that the cost (or, alternatively, the utility) of each individual be determined with respect to the current population so as to permit the broad ranking of the population. Individuals need only be rated better, similar, or worse than others, effectively allowing the decision maker to delay otherwise uninformed decisions until sufficient insight into the problem has been gained. At that point, the decision maker can adjust the current decision strategy, as the population evolves.

3 How do MOGAs differ from simple GAs?

In single-objective GAs, individual performance, as measured by the objective function, and individual fitness are so closely related that the objective function is sometimes referred to as the fitness function. The two are, however, not the same. In fact, whereas the objective function characterizes the problem and cannot be changed at will, assigned fitness is a direct measure of individual reproductive ability (i.e., expected number of offspring), forming an integral part of the GA search strategy.

This distinction becomes all the more important when performance is measured in terms of a vector of objective values, because fitness must remain a scalar. In this case, fitness assignment is a more elaborate process. For the sake of generality, the necessary scalarization of the objective vectors may be viewed as a multicriterion decision making problem involving a (finite) number of candidates, the individuals in the population [2, 6]. Individuals are thus assigned a measure of their utility indicating whether they perform better, worse, or similar to others, and possibly also how much better or worse. If the utility measure conveys only ordinal information, then fitness must be assigned through ranking. Otherwise, ranking or proportional fitness assignment may be used. This setup is general enough to include problems where individual performance must be assessed through pairwise comparison [7], such as when evolving game-playing programs.

Since the solution of a multicriterion decision making problem depends only on the vectorial performance of the available candidates and on the preferences of the decision maker, and not on any subsequent search or optimization, utility is also essentially different from fitness. In particular, techniques such as sharing affect the individuals' fitness, but not their utility or cost.

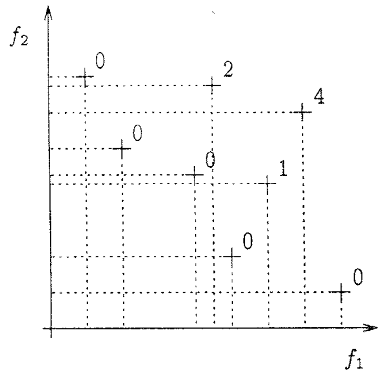

Figure 1: Pareto ranking

3.1 Scale-independent decision strategies

In the total absence of information concerning the relative importance of the objectives, Pareto-dominance is the only basis on which an individual can be said to perform better than another. Therefore, non-dominated individuals must all be considered best performers, and thus be assigned the same cost [1], e.g., zero. Deciding on the performance of dominated individuals is a more subjective matter. One may, for example, assign them a cost proportional to how many individuals in the population dominate them (Figure 1), in which case non-dominated individuals would also be treated as desired. This is essentially the Pareto-ranking scheme proposed in [2].

Another popular Pareto-ranking scheme [1], also known as non-dominated sorting [8], consists of removing the non-dominated individuals (still ranked zero, for ease of comparison) from contention, finding the non-dominated individuals in the remaining population and assigning them rank 1, and so forth, until the whole population is ranked. Both of these approaches guarantee that non-dominated individuals are all ranked best, and that individuals are consistently assigned better ranks than those they dominate. However, the first ranking scheme does appear to be easier to interpret and analyze mathematically [6].

When goal and/or priority information

is available for the objectives, it may become possible to discriminate between some non-dominated solutions. For example, if degradation in objective components which meet their goals does not go beyond the goal boundaries, and results in the improvement of objective components which do not yet satisfy the corresponding goals, then it should be accepted. Similarly, in a dual priority setup [6], it is only important to improve on high priority objectives (i.e., constraints) until the corresponding goals are met, after which improvement should be sought for the remaining objectives. These considerations have been formalized by the authors in terms of a transitive relational operator (*preferability*), based on Pareto-dominance, but which selectively excludes objectives according to their priority and to whether or not they meet their goals.

For simplicity, only one level of priority will be considered here. The full, multiple priority version of the preferability operator is described in detail in [6]. Consider two objective vectors $\mathbf{u}$ and $\mathbf{v}$ and a goal vector $\mathbf{g}$. Also, let the smile $\overset{u}{\smile}$ and the frown $\overset{u}{\frown}$ denote the components of $\mathbf{u}$ which meet their goals and those which do not, respectively. Assuming minimization, one can write

$$\mathbf{u}^{\overset{u}{\smile}} \le \mathbf{g}^{\overset{u}{\smile}} \quad \wedge \quad \mathbf{u}^{\overset{u}{\frown}} > \mathbf{g}^{\overset{u}{\frown}},$$

where the inequalities apply componentwise. This is equivalent to

$$\forall i \in \overset{u}{\smile},\ u_i \le g_i \quad \wedge \quad \forall i \in \overset{u}{\frown},\ u_i > g_i$$

where u_i and g_i represent the components of $\mathbf{u}$ and $\mathbf{g}$, respectively. Then, $\mathbf{u}$ is said to be preferable to $\mathbf{v}$ given $\mathbf{g}$ if and only if

$$(\mathbf{u}^{\overset{u}{\frown}} \;{}_{p}{<}\; \mathbf{v}^{\overset{u}{\frown}}) \vee \Big\{ (\mathbf{u}^{\overset{u}{\frown}} = \mathbf{v}^{\overset{u}{\frown}}) \wedge \Big[(\mathbf{v}^{\overset{u}{\smile}} \not\le \mathbf{g}^{\overset{u}{\smile}}) \vee (\mathbf{u}^{\overset{u}{\smile}} \;{}_{p}{<}\; \mathbf{v}^{\overset{u}{\smile}}) \Big] \Big\}$$

where $\mathbf{a} \;{}_{p}{<}\; \mathbf{b}$ denotes $\mathbf{a}$ dominates $\mathbf{b}$. In other words, $\mathbf{u}$ will be preferable to $\mathbf{v}$ if and only if one of the following is true:

1. The violating components of $\mathbf{u}$ dominate the corresponding components of $\mathbf{v}$.
2. The violating components of $\mathbf{u}$ are equal to the corresponding components of $\mathbf{v}$, but $\mathbf{v}$ violates at least another goal.
3. The violating components of $\mathbf{u}$ are equal to the corresponding components of $\mathbf{v}$, but $\mathbf{u}$ dominates $\mathbf{v}$ as a whole.

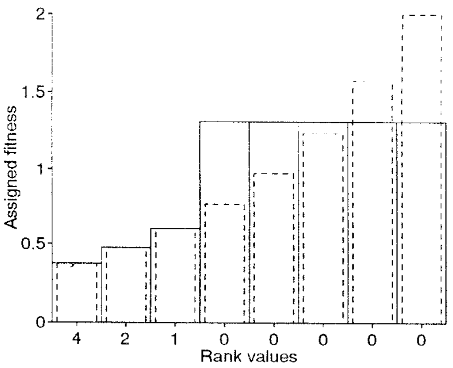

Figure 2: Rank-based fitness assignment

Like Pareto-dominance, this relation can be used to rank the individuals in a population by one of the methods described above.

3.2 Cost to fitness mapping and selection

Once cost has been assigned, selection can proceed in much the usual way. Suitable alternatives include rank-based cost to fitness mapping [9] followed by stochastic universal sampling [10] (or even roulette wheel selection) and tournament selection also based on cost, as reported in [11, 12].

Exponential rank-based fitness assignment is illustrated in Figure 2. Individuals are sorted by cost (the values are those from Figure 1) and first assigned fitness values according to an exponential rule (narrower bars). Then, a single value of fitness is derived for each group of individuals with the same cost, through averaging (wider bars).

3.3 Sharing

Although all preferred individuals are assigned the same fitness, their actual number of offspring, which must obviously be an integer, may differ. The imbalance can easily accumulate with the generations and result in the population drifting towards an arbitrary region of the the trade-off surface, a phenomenon known as genetic drift [13]. In addition, recombination and mutation may be less likely to produce individuals in certain regions of the trade-off surface (e.g., the

Fitness sharing	Kernel density estimation
Sharing function	Kernel function
Niche size (σ_{share})	Smoothing parameter (h)
Niche count	Density estimate

Table 1: The analogy between sharing and kernel estimation

extremes) than in others, causing the population to cover only a small part of it.

Fitness sharing [14], originally introduced to promote the sampling of multiple fitness peaks, helps counteract genetic drift by penalizing individuals due to the presence of other individuals in their neighbourhood. The *niche count* of each individual is initially set to zero and then incremented by a certain amount for every individual in the population, including itself. The contribution of an individual to another's niche count is dictated by a *sharing function*, which is a function of their mutual distance in genotypic, phenotypic, or objective space. Raw fitness values are then weighted by the inverse of the niche counts and subsequently normalized by the sum of the weights, before selection. In this way, the total fitness in the population is re-distributed (and thus shared) by the individuals. Fitness can also be shared only between individuals with the same raw fitness, by computing partial weight totals and performing the normalization within each group of such individuals [2].

The use of fitness sharing has been restricted by the difficulty found in determining the appropriate niche size, i.e., how close together individuals should be for degradation to occur. Current guidelines either make assumptions about the number and distribution of peaks in the cost landscape [3], or rely on the estimation of the (maximum) size of the trade-off surface based on the properties of the Pareto-set [2].

However, niche count computation (explained above) turns out to be remarkably similar to the kernel density estimation methods [4] known to statisticians. Basically, density estimates are computed in exactly the same way as niche counts, except for a constant factor. The parallel is drawn in Table 1. As in sharing, the choice of the smoothing parameter is ultimately subjective, but guidelines have been developed for certain kernels, such as the standard normal probability density function and the Epanechnikov kernel. The latter can be written as [4]

$$K_e(d/h) = \\ = \begin{cases} \frac{1}{2}c_n^{-1}(n+2)[1-(d/h)^2] & \text{if } d/h < 1 \\ 0 & \text{otherwise} \end{cases}$$

where n is the number of decision variables, c_n is the volume of the unit n-dimensional sphere and d/h is the normalized Euclidean distance between individuals. The parameter h is the smoothing parameter analogous to σ_{share}. Note that this kernel is, apart from the constant factor $\frac{1}{2}c_n^{-1}(n+2)$, a particular case of the family of power law sharing functions proposed by Goldberg and Richardson [14].

According to Silverman [4], a good choice (approximately optimal in the least mean integrated squared error sense if the population follows a multivariate normal distribution) of the smoothing parameter for the Epanechnikov kernel $K_e(d)$ is

$$h = \left[8c_n^{-1}(n+4)(2\sqrt{\pi})^n/N\right]^{1/(n+4)}$$

for a population with N individuals and identity covariance matrix. Populations with arbitrary sample covariance matrix S can simply be 'sphered' (or normalized) by multiplying each individual by a matrix R such that $RR^{\mathrm{T}} = S^{-1}$. This implies that the niche size (which depends on h and S) can be constantly, and automatically, adapted to suit the population at each generation, regardless of what the cost function may be.

These results can be used directly to perform sharing in Euclidean decision variable spaces. It might be possible to develop guidelines based on the same principles for other types of spaces.

3.4 Mating restriction

Mating restriction consists of biasing the way in which individuals are paired for recombination [3]. As the population distributes itself along the trade-off surface, recombining arbitrary pairs of individuals may conduce to the formation of a large number of unfit offspring, or lethals. To address this issue, mating can be restricted, where possible, to individuals within a given distance σ_{mate} from each other. Following the common practice of setting $\sigma_{\text{mate}} = \sigma_{\text{share}}$, individuals may be allowed to mate only if they lie within a distance h from each other in the 'sphered' space used for sharing.

3.5 Interactive optimization and changing environments

As the GA population evolves and trade-off information is acquired, the decision maker may wish to see the population concentrate on a smaller region of the trade-off surface, or even back off and move on to a totally different region. This can be achieved simply by changing the goals supplied to the GA at the cost assignment stage, which in turn affects the ranking of the population and modifies the cost landscape. The GA must then be able to respond quickly to such preference changes.

Introducing a small percentage (10–20%) of random individuals at each generation has been shown to make the GA more responsive to sudden changes in the fitness landscape [15]. This technique can be easily incorporated in a multiobjective GA.

4 Putting it all together

The implementation of a multiobjective GA incorporating the techniques described in the previous section will now be considered. Matlab [16] pseudo-code for a simple, aggregating, GA is given in Figure 3. Individual chromosomes (the rows of matrix `Chrom`) are initially generated at random, and then decoded, producing the corresponding vectors of decision variables, in matrix `DVar`. Evaluation is made in two steps: objective vectors are computed first (rows of `ObjV`), and then aggregated to produce a scalar measure of cost for each individual (the components of vector `Cost`). Fitness is assigned through ranking, with given selective pressure `SP`. Individuals are selected using SUS (stochastic universal sampling), recombined and mutated, and a new generation begins. Functions **`multobjfun`** and **`aggregate`** are written by the user, the former defining the problem and the latter implementing a fixed decision strategy, such as a weighted sum. The remaining functions implement the GA itself, and are not far from those found in the current version of the GA Toolbox for Matlab [17].

Preference-based multiobjective ranking (**`rank_prf`** in Figure 4) comes as a drop in replacement for **`aggregate`** which may take two optional parameters: a goal vector, **`GoalV`**, and a vector indicating the priority

```
Chrom = creatpop(NIND, LIND);

while Gen < MAXGEN
  DVar  = decode(Chrom);
  ObjV  = multobjfun(DVar);
  Cost  = aggregate(ObjV);
  Fitn  = ranking(Cost, SP);

  Ix    = sus(Fitn);
  SelCh = Chrom(Ix, :);
  SelCh = xover(SelCh, XOVR);
  Chrom = mutate(SelCh, MUTR);

  Gen = Gen + 1;
end
```

Figure 3: A simple aggregating GA

of the objectives, `PriorV` (not used in the example).

Niche counts `ShrC` are computed using a kernel estimator based on the Epanechnikov kernel. `DVar` is passed to the function twice because it constitutes simultaneously the sample data and the points where the population density needs to be estimated. The estimation function also returns the default smoothing parameter `Sigma` (h) and a matrix `R` such that `DVar*R` has identity covariance matrix, both of which are used at a later stage in mating restriction. The **`ranking`** function now uses `ShrC` to perform sharing between individuals with equal cost as an integral part of the fitness assignment procedure.

Since a small number `NImmigr` of individuals in the new population will consist of random immigrants, only `NIND-NImmigr` individuals are selected from the old population. Mating restriction is implemented by reordering the individuals in the population so that consecutive pairs of chromosomes in `SelCh` correspond, where possible, to individuals within a required distance `Sigma` of each other in normalized decision variable space. (The parental population is rotated and scaled according to the same transformation `R` used for niche count computation). The random immigrants are appended to the population after mutation, having to survive selection before being allowed to recombine. This will be most likely whenever the fitness landscape changes and the GA population is no longer adapted to it.

As can be easily seen, the only additional GA parameter in this second version of the

```
Chrom = creatpop(NIND, LIND);

while Gen < MAXGEN
  DVar    = decode(Chrom);
  ObjV    = multobjfun(DVar);
  Cost    = rank_prf(ObjV,GoalV);
  [ShrC, Sigma, R] ...
          = epanechnikov(DVar,DVar);
  Fitn    = ranking(Cost, SP, ShrC);

  Ix      = sus(Fitn, NIND - NImmigr);
  SelCh   = Chrom(Ix, :);
  SelDV   = DVar(Ix, :);
  PermIx = pairup(SelDV * R, Sigma);
  SelCh   = SelCh(PermIx, :);
  SelCh   = xover(SelCh, XOVR);
  Chrom   = [ mutate(SelCh, MUTR);
              creatpop(NImmigr, LIND) ];

  Gen = Gen + 1;
end
```

Figure 4: A multiobjective GA

GA is the number of random immigrants to be inserted in the population at each generation, the setting of which is not critical. Random immigrants make the GA more exploratory and thus more responsive to sudden preference changes, as long as a balanced amount of exploitation can still be maintained. In particular, selective pressure should probably be increased slightly, to compensate for the fact that the insertion of random immigrants into the population reduces the expected number of offspring of the best individual by NImmigr/NIND.

5 Experimental results

Several applications of multiobjective GAs have been reported in the literature, mainly related to control engineering. In an independent study, Whidborne *et al.* [18] have recently compared a multiobjective GA based on the preferability relation to other interactive multiobjective approaches such as the Method of Inequalities, and noted the tendency for the MOGA to produce solutions very similar to each other. However, they also pointed out that the GA did not include sharing or mating restriction.

To show how sharing and mating restriction together can significantly contribute to the performance of the GA, consider the minimization of the following two objectives:

$$f_1(x_1,\ldots,x_n) = 1 - \exp\left(-\sum_{i=1}^{n}(x_i - 1/\sqrt{n})^2\right)$$

$$f_2(x_1,\ldots,x_n) = 1 - \exp\left(-\sum_{i=1}^{n}(x_i + 1/\sqrt{n})^2\right)$$

which are defined for any number of decision variables n. The minimum of f_1 is located at $(x_1,\ldots,x_n) = (1/\sqrt{n},\ldots,1/\sqrt{n})$ for all n, and that of f_2 is located at $(x_1,\ldots,x_n) = (-1/\sqrt{n},\ldots,-1/\sqrt{n})$. Due to the symmetry of the two functions, the Pareto-optimal set clearly corresponds to all points on the line defined by

$$x_1 = x_2 = \cdots = x_n \wedge -1/\sqrt{n} \leq x_1 \leq 1/\sqrt{n}$$

A simple genetic algorithm with a population size of 100 individuals, binary chromosomes, reduced-surrogate shuffle crossover and binary mutation was used to approach this problem for $n = 8$. Decision variables were Gray-encoded as 16-bit strings in the interval $[-2, 2)$ and concatenated to form the chromosomes. Multiobjective ranking was performed as described and illustrated earlier in Figure 1.

Running this GA for 100 generations, without sharing or mating restriction, shows how the population tends to concentrate on a small region of the trade-off surface (Figure 5). Non-dominated individuals are marked with filled circles (•) and other individuals with empty circles (○). The solid line represents the best approximation to the real trade-off surface (dashed line) known as a consequence of the GA run.

If, however, sharing and mating restriction are implemented in the decision variable domain as in the example in Figure 4, the population is able to remain distributed across the whole trade-off surface. This can be seen in Figure 6.

The population can also be driven to sample a given region of the trade-off surface by setting goals accordingly. Figure 7 shows the distribution of the population after setting the goals and running the GA for another 50 generations. Most of the population can be seen to be concentrated in the preferred region of the trade-off surface, as desired.

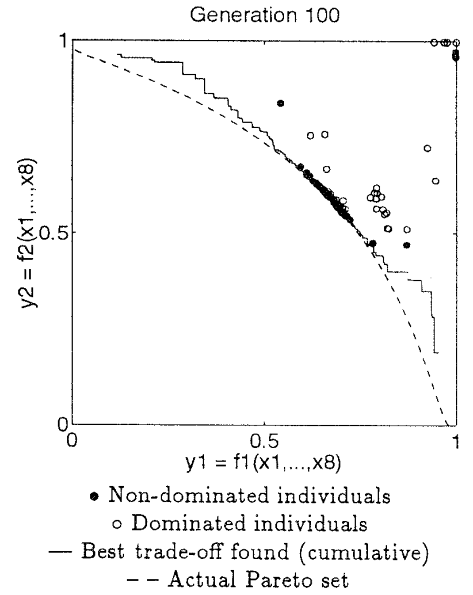

Figure 5: Multiobjective GA without sharing or mating restriction

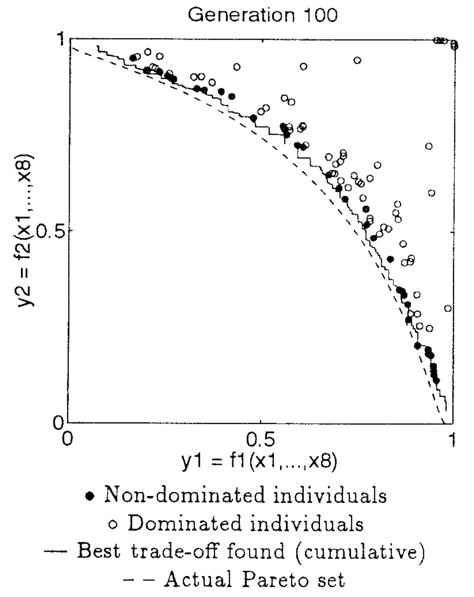

Figure 6: Multiobjective GA with sharing and mating restriction

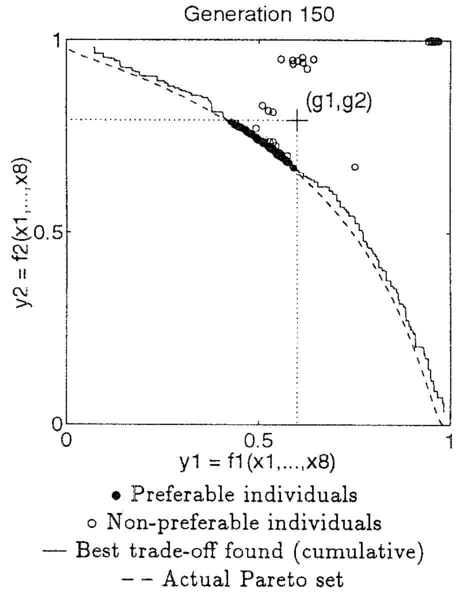

Figure 7: Zooming in on a region of the trade-off surface by setting goals accordingly

An application of the multiobjective GA described here in the design of controllers for gas turbine engines is reported in [19].

6 Concluding remarks

Although multiobjective genetic algorithms are still an area undergoing development, their application to real world problems is becoming increasingly feasible. In combination with a suitable graphical user interface, multiobjective GAs can become a powerful, and possibly interactive, decision support tool, allowing a decision maker to learn about the problem before committing to a final decision.

Due to the multi-solution nature of most multiobjective problems, fitness sharing is needed to maintain diversity in the population. However, guidelines on how to set the sharing parameter have been too dependent on suppositions about the fitness landscape which are difficult to make in any practical setting. Understanding sharing as something similar to density estimation can make the use of sharing, and thus that of multiobjective GAs, more practical. More elaborate density estimation techniques, such as adaptive kernel density estimation [4], may further improve the quality of sharing. On the other hand, nearest-neighbour estimators may be easier to extend to non-Euclidean spaces, and thus be more appropriate to ordering and grouping problems, for example.

Finally, multiobjective evolutionary optimization is a much broader area than reported here, and the interested reader is referred to [20] for an up-to-date overview.

Acknowledgement The authors wish to acknowledge the support of the UK Engineering and Physical Sciences Research Council (Grant GR/J70857).

References

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, 1989.

[2] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Genetic Algorithms: Proc. Fifth International Conference* (S. Forrest, ed.), pp. 416–423, Morgan Kaufmann, 1993.

[3] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proc. Third International Conference on Genetic Algorithms* (J. D. Schaffer, ed.), pp. 42–50, Morgan Kaufmann, 1989.

[4] B. W. Silverman, *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, 1986.

[5] A. Ben-Tal, "Characterization of Pareto and lexicographic optimal solutions," in *Multiple Criteria Decision Making Theory and Application* (G. Fandel and T. Gal, eds.), vol. 177 of *Lecture Notes in Economics and Mathematical Systems*, pp. 1–11, 1980.

[6] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms I: A unified formulation," Research report 564, Dept. Automatic Control and Systems Eng., University of Sheffield, U.K., Jan. 1995.

[7] V. R. R. Uppuluri, "Prioritization techniques based on stochastic paired comparisons," in *Multiple Criteria Decision Making and Risk Analysis Using Microcomputers* (B. Karpak and S. Zionts, eds.), pp. 293–303, Springer-Verlag, 1989.

[8] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, Fall 1994. To appear.

[9] J. E. Baker, "Adaptive selection methods for genetic algorithms," in *Proc. First International Conference on Genetic Algorithms* (J. J. Grefenstette, ed.), pp. 101–111, Lawrence Erlbaum, 1985.

[10] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in Grefenstette [21], pp. 14–21.

[11] S. E. Cieniawski, "An investigation of the ability of genetic algorithms to generate the tradeoff curve of a multiobjective groundwater monitoring problem," Master's thesis, University of Illinois at Urbana-Champaign, 1993.

[12] B. J. Ritzel, J. W. Eheart, and S. Ranjithan, "Using genetic algorithms to solve a multiple objective groundwater pollution containment problem," *Water Resources Research*, vol. 30, pp. 1589–1603, May 1994.

[13] D. E. Goldberg and P. Segrest, "Finite markov chain analysis of genetic algorithms," in Grefenstette [21], pp. 1–8.

[14] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in Grefenstette [21], pp. 41–49.

[15] J. J. Grefenstette, "Genetic algorithms for changing environments," in *Parallel Problem Solving from Nature, 2* (R. Männer and B. Manderick, eds.), pp. 137–144, North-Holland, 1992.

[16] The MathWorks, Inc., MATLAB *Reference Guide*, Aug. 1992.

[17] A. Chipperfield, P. Fleming, H. Pohlheim, and C. Fonseca, "Genetic algorithm toolbox user's guide," Research report 512, Dept. Automatic Control and Systems Eng., University of Sheffield, U.K., July 1994.

[18] J. F. Whidborne, D.-W. Gu, and I. Postlethwaite, "Algorithms for the method of inequalities – a comparative study," in *Proc. American Control Conference*, 1995.

[19] A. J. Chipperfield and P. J. Fleming, "Gas turbine engine controller design using multiobjective genetic algorithms," in *Submitted to GALESIA*, 1995.

[20] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, Spring 1995. To appear.

[21] J. J. Grefenstette, ed., *Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum, 1987.