Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization*

Carlos M. Fonseca† and Peter J. Fleming‡
Dept. Automatic Control and Systems Eng.
University of Sheffield
Sheffield S1 4DU, U.K.

Abstract

The paper describes a rank-based fitness assignment method for Multiple Objective Genetic Algorithms (MOGAs). Conventional niche formation methods are extended to this class of multimodal problems and theory for setting the niche size is presented. The fitness assignment method is then modified to allow direct intervention of an external decision maker (DM). Finally, the MOGA is generalised further: the genetic algorithm is seen as the optimizing element of a multiobjective optimization loop, which also comprises the DM. It is the interaction between the two that leads to the determination of a satisfactory solution to the problem. Illustrative results of how the DM can interact with the genetic algorithm are presented. They also show the ability of the MOGA to uniformly sample regions of the trade-off surface.

1 INTRODUCTION

Whilst most real world problems require the simultaneous optimization of multiple, often competing, criteria (or objectives), the solution to such problems is usually computed by combining them into a single criterion to be optimized, according to some *utility function*. In many cases, however, the utility function is not well known prior to the optimization process. The whole problem should then be treated as a *multiobjective problem* with *non-commensurable* objectives. In this way, a number of solutions can be found which provide the decision maker (DM) with insight into the characteristics of the problem before a final solution is chosen.

*in *Genetic Algorithms: Proceedings of the Fifth International Conference* (S. Forrest, ed.), San Mateo, CA: Morgan Kaufmann, July 1993.

†`C.Fonseca@shef.ac.uk`

‡`P.Fleming@shef.ac.uk`

Multiobjective optimization (MO) seeks to optimize the components of a vector-valued cost function. Unlike single objective optimization, the solution to this problem is not a single point, but a family of points known as the Pareto-optimal set. Each point in this surface is optimal in the sense that no improvement can be achieved in one cost vector component that does not lead to degradation in at least one of the remaining components. Assuming, without loss of generality, a minimization problem, the following definitions apply:

Definition 1 (inferiority)
A vector $\mathbf{u} = (u_1, \ldots, u_n)$ *is said to be inferior to* $\mathbf{v} = (v_1, \ldots, v_n)$ *iff* $\mathbf{v}$ *is partially less than* $\mathbf{u}$ ($\mathbf{v}\ p\!<\ \mathbf{u}$), *i.e.,*

$$\forall i = 1, \ldots, n\,,\ v_i \le u_i \quad \wedge \quad \exists i = 1, \ldots, n : v_i < u_i$$

Definition 2 (superiority)
A vector $\mathbf{u} = (u_1, \ldots, u_n)$ *is said to be superior to* $\mathbf{v} = (v_1, \ldots, v_n)$ *iff* $\mathbf{v}$ *is inferior to* $\mathbf{u}$.

Definition 3 (non-inferiority)
Vectors $\mathbf{u} = (u_1, \ldots, u_n)$ *and* $\mathbf{v} = (v_1, \ldots, v_n)$ *are said to be non-inferior to one another if* $\mathbf{v}$ *is neither inferior nor superior to* $\mathbf{u}$.

Each element in the Pareto-optimal set constitutes a non-inferior solution to the MO problem. Non-inferior solutions have been obtained by solving appropriately formulated NP problems, on a one at a time basis. Methods used include the weighted sum approach, the ε-constraint method and goal programming. Within the goal programming category, the goal attainment method has shown to be particularly useful in Computer Aided Control System Design (CACSD) (Fleming, 1985; Farshadnia, 1991; Fleming *et al.*, 1992). Multicriteria target vector optimization has recently been used in combination with genetic algorithms (Wienke *et al.*, 1992).

By maintaining a population of solutions, genetic algorithms can search for many non-inferior solutions in parallel. This characteristic makes GAs very attractive for solving MO problems.

2 VECTOR EVALUATED GENETIC ALGORITHMS

Being aware of the potential GAs have in multiobjective optimization, Schaffer (1985) proposed an extension of the simple GA (SGA) to accommodate vector-valued fitness measures, which he called the Vector Evaluated Genetic Algorithm (VEGA). The selection step was modified so that, at each generation, a number of sub-populations was generated by performing proportional selection according to each objective function in turn. Thus, for a problem with q objectives, q sub-populations of size N/q each would be generated, assuming a population size of N. These would then be shuffled together to obtain a new population of size N, in order for the algorithm to proceed with the application of crossover and mutation in the usual way.

However, as noted by Richardson *et al.* (1989), shuffling all the individuals in the sub-populations together to obtain the new population is equivalent to linearly combining the fitness vector components to obtain a single-valued fitness function. The weighting coefficients, however, depend on the current population.

This means that, in the general case, not only will two non-dominated individuals be sampled at different rates, but also, in the case of a concave trade-off surface, the population will tend to split into different species, each of them particularly strong in one of the objectives. Schaffer anticipated this property of VEGA and called it *speciation*. Speciation is undesirable in that it is opposed to the aim of finding a compromise solution.

To avoid combining objectives in any way requires a different approach to selection. The next section describes how the concept of inferiority alone can be used to perform selection.

3 A RANK-BASED FITNESS ASSIGNMENT METHOD FOR MOGAs

Consider an individual x_i at generation t which is dominated by $p_i^{(t)}$ individuals in the current population. Its current position in the individuals' rank can be given by

$$\mathrm{rank}(x_i, t) = 1 + p_i^{(t)}.$$

All non-dominated individuals are assigned rank 1, see Figure 1. This is not unlike a class of selection methods proposed by Fourman (1985) for constrained optimization, and correctly establishes that the individual labelled 3 in the figure is worse than individual labelled 2, as the latter lies in a region of the trade-off which is less well described by the remaining individuals. The

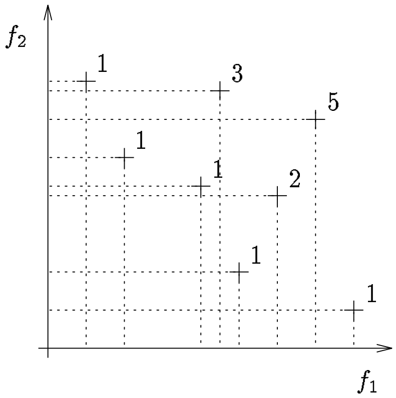

Figure 1: Multiobjective Ranking

method proposed by Goldberg (1989, p. 201) would treat these two individuals indifferently.

Concerning fitness assignment, one should note that not all ranks will necessarily be represented in the population at a particular generation. This is also shown in the example in Figure 1, where rank 4 is absent. The traditional assignment of fitness according to rank may be extended as follows:

1. Sort population according to rank.
2. Assign fitnesses to individuals by interpolating from the best (rank 1) to the worst (rank $n^* \leq N$) in the usual way, according to some function, usually linear but not necessarily.
3. Average the fitnesses of individuals with the same rank, so that all of them will be sampled at the same rate. Note that this procedure keeps the global population fitness constant while maintaining appropriate selective pressure, as defined by the function used.

The fitness assignment method just described appears as an extension of the standard assignment of fitness according to rank, to which it maps back in the case of a single objective, or that of non-competing objectives.

4 NICHE-FORMATION METHODS FOR MOGAs

Conventional fitness sharing techniques (Goldberg and Richardson, 1987; Deb and Goldberg, 1989) have been shown to be to effective in preventing genetic drift, in multimodal function optimization. However, they introduce another GA parameter, the niche size σ_{share}, which needs to be set carefully. The existing theory for setting the value of σ_{share} assumes that the solution set is composed by an *a priori* known finite number of peaks and uniform niche placement. Upon convergence, local optima are occupied by a number of individuals proportional to their fitness values.

On the contrary, the global solution of an MO problem is flat in terms of individual fitness, and there is no way of knowing the size of the solution set beforehand, in terms of a phenotypic metric. Also, local optima are generally not interesting to the designer, who will be more concerned with obtaining a set of globally non-dominated solutions, possibly uniformly spaced and illustrative of the global trade-off surface. The use of ranking already forces the search to concentrate only on global optima. By implementing fitness sharing in the *objective value* domain rather than the decision variable domain, and only between pairwise non-dominated individuals, one can expect to be able to evolve a uniformly distributed representation of the global trade-off surface.

Niche counts can be consistently incorporated into the extended fitness assignment method described in the previous section by using them to scale individual fitnesses within each rank. The proportion of fitness allocated to the set of currently non-dominated individuals as a whole will then be independent of their sharing coefficients.

4.1 CHOOSING THE PARAMETER σ_{share}

The sharing parameter σ_{share} establishes how far apart two individuals must be in order for them to decrease each other's fitness. The exact value which would allow a number of points to sample a trade-off surface only tangentially interfering with one another obviously depends on the area of such a surface.

As noted above in this section, the size of the set of solutions to a MO problem expressed in the decision variable domain is not known, since it depends on the objective function mappings. However, when expressed in the objective value domain, and due to the definition of non-dominance, an upper limit for the size of the solution set can be calculated from the minimum and maximum values each objective assumes within that set. Let S be the solution set in the decision variable domain, $\mathbf{f}(S)$ the solution set in the objective domain and $\mathbf{y} = (y_1, \ldots, y_q)$ any objective vector in $\mathbf{f}(S)$. Also, let

$$\begin{aligned} \mathbf{m} &= (\min_{\mathbf{y}} y_1, \ldots, \min_{\mathbf{y}} y_q) = (m_1, \ldots, m_q) \\ \mathbf{M} &= (\max_{\mathbf{y}} y_1, \ldots, \max_{\mathbf{y}} y_q) = (M_1, \ldots, M_q) \end{aligned}$$

as illustrated in Figure 2. The definition of trade-off surface implies that any line parallel to any of the axes will have not more than one of its points in $\mathbf{f}(S)$, which eliminates the possibility of it being rugged, i.e., each objective is a single-valued function of the remaining objectives. Therefore, the true area of $\mathbf{f}(S)$ will be less than the sum of the areas of its projections according to each of the axes. Since the maximum area of each projection will be at most the area of the corresponding face of the hyperparallelogram defined by $\mathbf{m}$ and

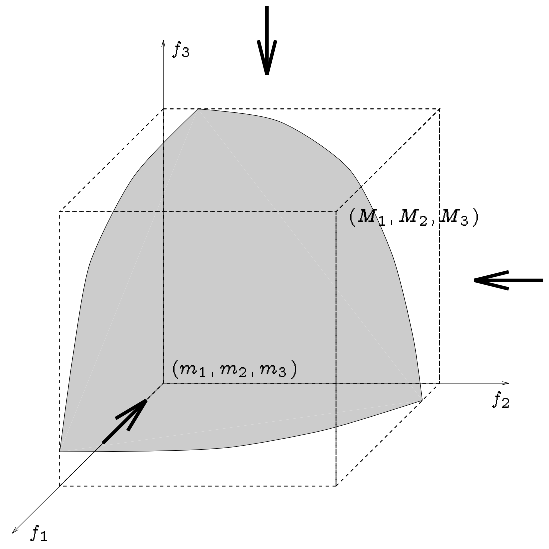

Figure 2: An Example of a Trade-off Surface in 3-Dimensional Space

$\mathbf{M}$, the hyperarea of $\mathbf{f}(S)$ will be less than

$$A = \sum_{i=1}^{q} \prod_{\substack{j=1 \\ j \neq i}}^{q} (M_j - m_j)$$

which is the sum of the areas of each different face of a hyperparallelogram of edges $(M_j - m_j)$ (Figure 3).

In accordance with the objectives being non-commensurable, the use of the ∞-norm for measuring the distance between individuals seems to be the most natural one, while also being the simplest to compute. In this case, the user is still required to specify an individual σ_{share} for each of the objectives. However, the metric itself does not combine objective values in any way.

Assuming that objectives are normalized so that all sharing parameters are the same, the maximum number of points that can sample area A without interfering with each other can be computed as the number of hypercubes of volume σ_{share}^q that can be placed over the hyperparallelogram defined by A (Figure 4). This can be computed as the difference in volume between two hyperparallelograms, one with edges $(M_i - m_i + \sigma_{\text{share}})$ and the other with edges $(M_i - m_i)$, divided by the volume of a hypercube of edge σ_{share}, i.e.

$$N = \frac{\prod_{i=1}^{q} (M_i - m_i + \sigma_{\text{share}}) - \prod_{i=1}^{q} (M_i - m_i)}{\sigma_{\text{share}}^q}$$

Conversely, given a number of individuals (points), N, it is now possible to estimate σ_{share} by solving the

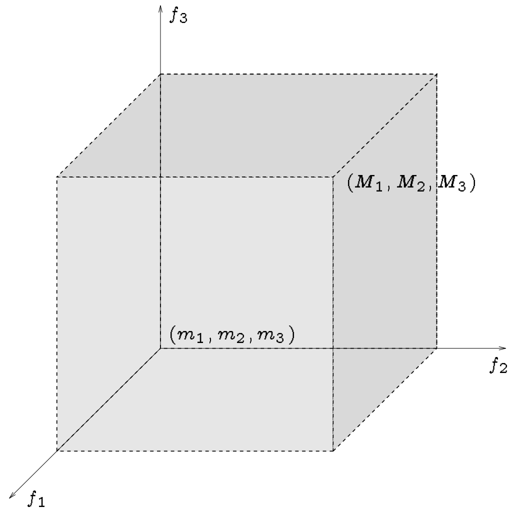

Figure 3: Upper Bound for the Area of a Trade-off Surface limited by the Parallelogram defined by (m_1, m_2, m_3) and (M_1, M_2, M_3)

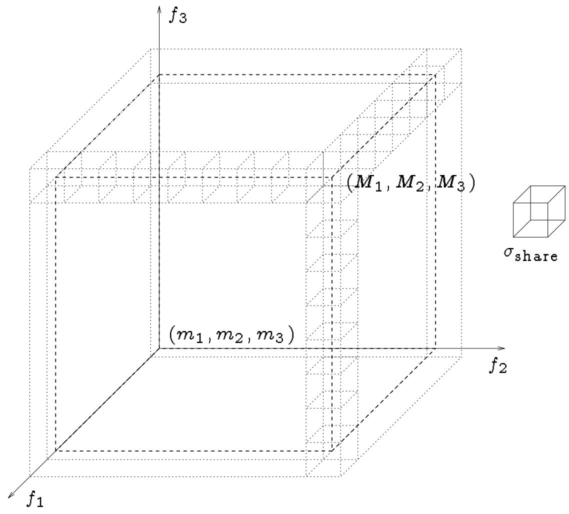

Figure 4: Sampling Area A. Each Point is σ_{share} apart from each of its Neighbours (∞-norm)

$(q-1)$-order polynomial equation

$$N\sigma_{\text{share}}^{q-1} - \frac{\prod_{i=1}^{q}(M_i - m_i + \sigma_{\text{share}}) - \prod_{i=1}^{q}(M_i - m_i)}{\sigma_{\text{share}}} = 0$$

for $\sigma_{\text{share}} > 0$.

4.2 CONSIDERATIONS ON MATING RESTRICTION AND CHROMOSOME CODING

The use of mating restriction was suggested by Goldberg in order to avoid excessive competition between distant members of the population. The ability to calculate σ_{share} on the objective domain immediately suggests the implementation of mating restriction schemes on the same domain, by defining the corresponding parameter, σ_{mating}.

Mating restriction assumes that neighbouring fit individuals are genotypically similar, so that they can form stable niches. Extra attention must therefore be paid to the coding of the chromosomes. Gray codes, as opposed to standard binary, are known to be useful for their property of adjacency. However, the coding of decision variables as the concatenation of independent binary strings cannot be expected to consistently express any relationship between them.

On the other hand, the Pareto set, when represented in the decision variable domain, will certainly exhibit such dependencies. In that case, even relatively small regions of the Pareto-set may not be characterized by a single, high-order, schema and the ability of mating restriction to reduce the formation of lethals will be considerably diminished. As the size of the solution set increases, an increasing number of individuals is necessary in order to assure niche sizes small enough for the individuals within each niche to be sufficiently similar to each other.

Given a reduced number of individuals, the Pareto set of a given vector function may simply be too large for this to occur. Since, on the other hand, the designer is often looking for a single compromise solution to the MO problem, reducing the size of the solution set by deciding at a higher level which individuals express a good compromise would help to overcome the problems raised above.

5 INCORPORATING HIGHER-LEVEL DECISION MAKING IN THE SELECTION ALGORITHM

When presented with the trade-off surface for a given function, the decision maker (DM) would have to decide which of all of the non-dominated points to choose as *the* solution to the problem. First, the regions of the Pareto set which express good compromises according to some problem-specific knowledge would be identified. Then, having a clearer picture of what is achievable, the idea of compromise would be refined until the solution was found. As a consequence, a very precise knowledge of the areas that end up being discarded is of doubtful utility. Only the “interesting” regions of the Pareto set need to be well known.

Reducing the size of the solution set calls for higher-level decision making to be incorporated in the selection algorithm. The idea is not to reduce the scope of the search, but simply to zoom in on the region of the

Pareto set of interest to the DM by providing external information to the selection algorithm.

The fitness assignment method described earlier was modified in order to accept such information in the form of goals to be attained, in a similar way to that used by the conventional goal attainment method (Gembicki, 1974), which will now be briefly introduced.

5.1 THE GOAL ATTAINMENT METHOD

The goal attainment method solves the multiobjective optimization problem defined as

$$\min_{\mathbf{x}\in\Omega} \mathbf{f}(\mathbf{x})$$

where $\mathbf{x}$ is the design parameter vector, Ω the feasible parameter space and $\mathbf{f}$ the vector objective function, by converting it into the following nonlinear programming problem:

$$\min_{\lambda,\mathbf{x}\in\Omega} \lambda$$

such that

$$f_i - w_i\lambda \le g_i$$

Here, g_i are goals for the design objectives f_i, and $w_i \ge 0$ are weights, all of them specified by the designer beforehand. The minimization of the scalar λ leads to the finding of a non-dominated solution which under- or over-attains the specified goals to a degree represented by the quantities $w_i\lambda$.

5.2 A MODIFIED MO RANKING SCHEME TO INCLUDE GOAL INFORMATION

The MO ranking procedure previously described was extended to accommodate goal information by altering the way in which individuals are compared with one another. In fact, degradation in vector components which meet their goals is now acceptable provided it results in the improvement of other components which do not satisfy their goals and it does not go beyond the goal boundaries. This makes it possible for one to prefer one individual to another even though they are both non-dominated. The algorithm will then identify and evolve the relevant region of the trade-off surface.

Still assuming a minimization problem, consider two q-dimensional objective vectors, $\mathbf{y}_a = (y_{a,1}, \ldots, y_{a,q})$ and $\mathbf{y}_b = (y_{b,1}, \ldots, y_{b,q})$, and the goal vector $\mathbf{g} = (g_1, \ldots, g_q)$. Also consider that $\mathbf{y}_a$ is such that it meets a number, $q-k$, of the specified goals. Without loss of generality, one can write

$$\exists\, k = 1, \ldots, q-1 : \forall\, i = 1, \ldots, k\,,\quad \forall\, j = k+1, \ldots, q\,,\ (y_{a,i} > g_i) \wedge (y_{a,j} \le g_j) \qquad \text{(A)}$$

which assumes a convenient permutation of the objectives. Eventually, $\mathbf{y}_a$ will meet none of the goals, i.e.,

$$\forall\, i = 1, \ldots, q\,,\ (y_{a,i} > g_i) \qquad \text{(B)}$$

or even all of them, and one can write

$$\forall\, j = 1, \ldots, q\,,\ (y_{a,j} \le g_j) \qquad \text{(C)}$$

In the first case (A), $\mathbf{y}_a$ meets goals $k+1, \ldots, q$ and, therefore, will be *preferable to* $\mathbf{y}_b$ simply if it dominates $\mathbf{y}_b$ with respect to its first k components. For the case where all of the first k components of $\mathbf{y}_a$ are equal to those of $\mathbf{y}_b$, $\mathbf{y}_a$ will still be *preferable to* $\mathbf{y}_b$ if it dominates $\mathbf{y}_b$ with respect to the remaining components, or if the remaining components of $\mathbf{y}_b$ do not meet all their goals. Formally, $\mathbf{y}_a$ will be *preferable to* $\mathbf{y}_b$, if and only if

$$\begin{aligned}&\left(\mathbf{y}_{a,(1,\ldots,k)}\ p\!<\ \mathbf{y}_{b,(1,\ldots,k)}\right) \vee \\ &\quad \left\{ \left(\mathbf{y}_{a,(1,\ldots,k)} = \mathbf{y}_{b,(1,\ldots,k)}\right) \wedge \right. \\ &\qquad \left[\left(\mathbf{y}_{a,(k+1,\ldots,q)}\ p\!<\ \mathbf{y}_{b,(k+1,\ldots,q)}\right) \vee \right. \\ &\qquad\qquad \left.\left. \sim\left(\mathbf{y}_{b,(k+1,\ldots,q)} \le \mathbf{g}_{(k+1,\ldots,q)}\right)\right] \right\}\end{aligned}$$

In the second case (B), $\mathbf{y}_a$ satisfies none of the goals. Then, $\mathbf{y}_a$ is *preferable to* $\mathbf{y}_b$ if and only if it dominates $\mathbf{y}_b$, i.e.,

$$\mathbf{y}_a\ p\!<\ \mathbf{y}_b$$

Finally, in the third case (C) $\mathbf{y}_a$ meets all of the goals, which means that it is a satisfactory, though not necessarily optimal, solution. In this case, $\mathbf{y}_a$ is *preferable to* $\mathbf{y}_b$, if and only if it dominates $\mathbf{y}_b$ or $\mathbf{y}_b$ is not satisfactory, i.e.,

$$(\mathbf{y}_a\ p\!<\ \mathbf{y}_b) \vee \sim(\mathbf{y}_b \le \mathbf{g})$$

The use of the relation *preferable to* as just described, instead of the simpler relation *partially less than*, implies that the solution set be delimited by those non-dominated points which tangentially achieve one or more goals. Setting all the goals to $\pm\infty$ will make the algorithm try to evolve a discretized description of the whole Pareto set.

Such a description, inaccurate though it may be, can guide the DM in refining its requirements. When goals can be supplied interactively at each GA generation, the decision maker can reduce the size of the solution set gradually while learning about the trade-off between objectives. The variability of the goals acts as a changing environment to the GA, and does not impose any constraints on the search space. Note that appropriate sharing coefficients can still be calculated as before, since the size of the solution set changes in a way which is known to the DM.

This strategy of progressively articulating the DM preferences, while the algorithm runs, to guide the search, is not new in operations research. The main disadvantage of the method is that it demands a higher effort from the DM. On the other hand, it potentially reduces the number of function evaluations required when compared to a method for *a posteriori* articulation of preferences, as well as providing less alternative

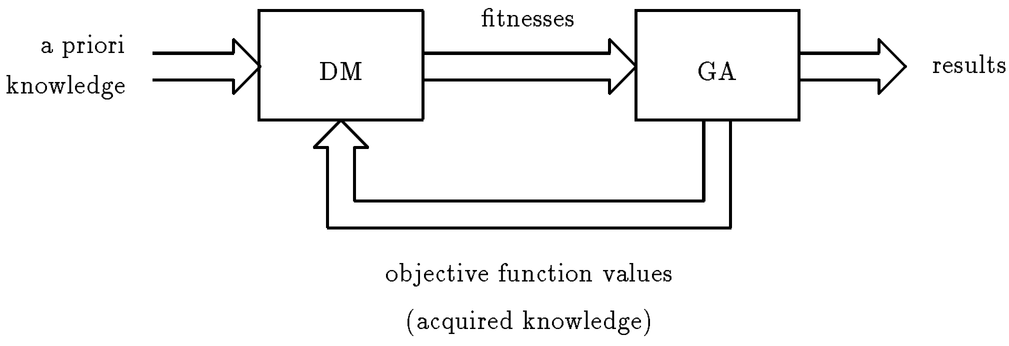

Figure 5: A General Multiobjective Genetic Optimizer

points at each iteration, which are certainly easier for the DM to discriminate between than the whole Pareto set at once.

6 THE MOGA AS A METHOD FOR PROGRESSIVE ARTICULATION OF PREFERENCES

The MOGA can be generalized one step further. The DM action can be described as the consecutive evaluation of some not necessarily well defined *utility function*. The utility function expresses the way in which the DM combines objectives in order to prefer one point to another and, ultimately, is the function which establishes the basis for the GA population to evolve. Linearly combining objectives to obtain a scalar fitness, on the one hand, and simply ranking individuals according to non-dominance, on the other, both correspond to two different attitudes of the DM. In the first case, it is assumed that the DM knows exactly what to optimize, for example, financial cost. In the second case, the DM is making no decision at all apart from letting the optimizer use the broadest definition of MO optimality. Providing goal information, or using sharing techniques, simply means a more elaborated attitude of the DM, that is, a less straightforward utility function, which may even vary during the GA process, but still just another utility function.

A multiobjective genetic optimizer would, in general, consist of a standard genetic algorithm presenting the DM at each generation with a set of points to be assessed. The DM makes use of the concept of Pareto optimality and of any a priori information available to express its preferences, and communicates them to the GA, which in turn replies with the next generation. At the same time, the DM learns from the data it is presented with and eventually refines its requirements until a suitable solution has been found (Figure 5).

In the case of a human DM, such a set up may require reasonable interaction times for it to become attractive. The natural solution would consist of speeding up the process by running the GA on a parallel architecture. The most appealing of all, however, would be the use of an automated DM, such as an expert system.

7 INITIAL RESULTS

The MOGA is currently being applied to the step response optimization of a Pegasus gas turbine engine. A full non-linear model of the engine (Hancock, 1992), implemented in SIMULINK (MathWorks, 1992b), is used to simulate the system, given a number of initial conditions and the controller parameter settings. The GA is implemented in MATLAB (MathWorks, 1992a; Fleming *et al.*, 1993), which means that all the code actually runs in the same computation environment.

The logarithm of each controller parameter was Gray encoded as a 14-bit string, leading to 70-bit long chromosomes. A random initial population of size 80 and standard two-point reduced surrogate crossover and binary mutation were used. The initial goal values were set according to a number of performance requirements for the engine. Four objectives were used:

- t_r The time taken to reach 70% of the final output change. Goal: $t_r \leq 0.59$s.
- t_s The time taken to settle within $\pm 10\%$ of the final output change. Goal: $t_s \leq 1.08$s.
- os Overshoot, measured relatively to the final output change. Goal: $os \leq 10\%$.
- err A measure of the output error 4 seconds after the step, relative to the final output change. Goal: $err \leq 10\%$.

During the GA run, the DM stores all non-dominated points evaluated up to the current generation. This constitutes acquired knowledge about the trade-offs available in the problem. From these, the relevant points are identified, the size of the trade-off surface estimated and σ_{share} set. At any time in the optimiza-

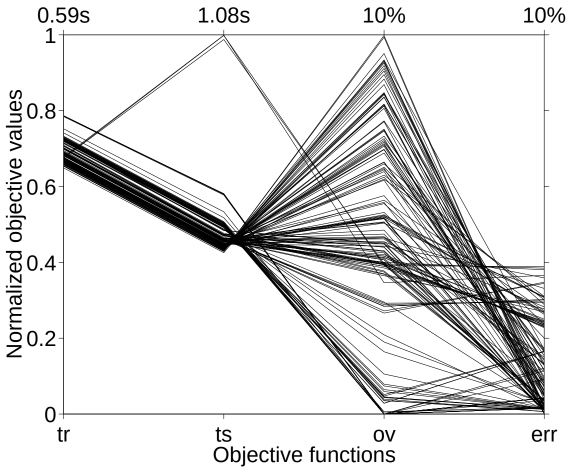

Figure 6: Trade-off Graph for the Pegasus Gas Turbine Engine after 40 Generations (Initial Goals)

tion process, the goal values can be changed, in order to zoom in on the region of interest.

A typical trade-off graph, obtained after 40 generations with the initial goals, is presented in Figure 6 and represents the accumulated set of satisfactory nondominated points. At this stage, the setting of a much tighter goal for the output error ($err \leq 0.1\%$) reveals the graph in Figure 7, which contains a subset of the points in Figure 6. Continuing to run the GA, more definition can be obtained in this area (Figure 8). Figure 9 presents an alternative view of these solutions, illustrating the arising step responses.

8 CONCLUDING REMARKS

Genetic algorithms, searching from a population of points, seem particularly suited to multiobjective optimization. Their ability to find global optima while being able to cope with discontinuous and noisy functions has motivated an increasing number of applications in engineering and related fields. The development of the MOGA is one expression of our wish to bring decision making into engineering design, in general, and control system design, in particular.

An important problem arising from the simple Pareto-based fitness assignment method is that of the global size of the solution set. Complex problems can be expected to exhibit a large and complex trade-off surface which, to be sampled accurately, would ultimately overload the DM with virtually useless information. Small regions of the trade-off surface, however, can still be sampled in a Pareto-based fashion, while the decision maker learns and refines its requirements. Niche formation methods are transferred to the objective value domain in order to take advantage of the properties of the Pareto set.

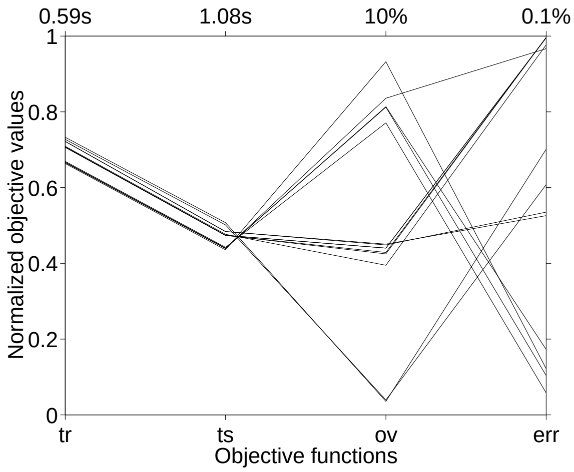

Figure 7: Trade-off Graph for the Pegasus Gas Turbine Engine after 40 Generations (New Goals)

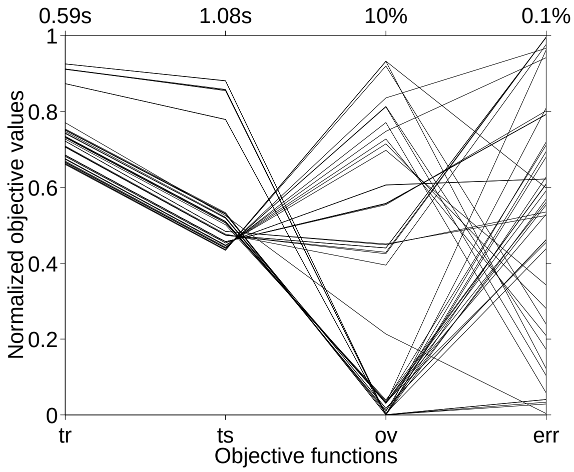

Figure 8: Trade-off Graph for the Pegasus Gas Turbine Engine after 60 Generations (New Goals)

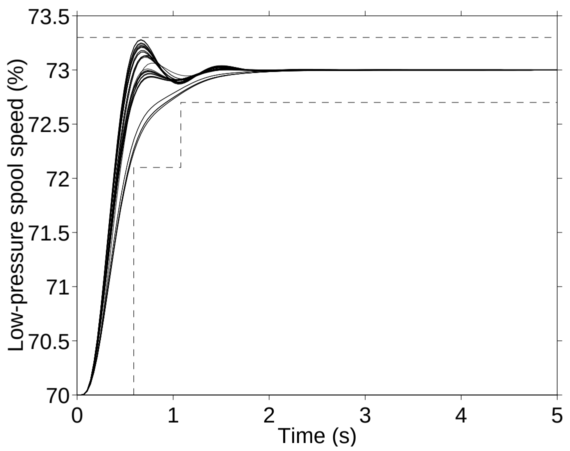

Figure 9: Satisfactory Step Responses after 60 Generations (New Goals)

Initial results, obtained from a real world engineering problem, show the ability of the MOGA to evolve uniformly sampled versions of trade-off surface regions. They also illustrate how the goals can be changed during the GA run.

Chromosome coding, and the genetic operators themselves, constitute areas for further study. Redundant codings would eventually allow the selection of the appropriate representation while evolving the trade-off surface, as suggested in (Chipperfield *et al.*, 1992). The direct use of real variables to represent an individual together with correlated mutations (Bäck *et al.*, 1991) and some clever recombination operator(s) may also be interesting. In fact, correlated mutations should be able to identify how decision variables relate to each other within the Pareto set.

Acknowledgements

The first author gratefully acknowledges support by Programa CIENCIA, Junta Nacional de Investigação Científica e Tecnológica, Portugal.

References

Bäck, T., Hoffmeister, F., and Schwefel, H.-P. (1991). A survey of evolution strategies. In Belew, R., editor, *Proc. Fourth Int. Conf. on Genetic Algorithms*, pp. 2–9. Morgan Kaufmann.

Chipperfield, A. J., Fonseca, C. M., and Fleming, P. J. (1992). Development of genetic optimization tools for multi-objective optimization problems in CACSD. In *IEE Colloq. on Genetic Algorithms for Control Systems Engineering*, pp. 3/1–3/6. The Institution of Electrical Engineers. Digest No. 1992/106.

Deb, K. and Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. In Schaffer, J. D., editor, *Proc. Third Int. Conf. on Genetic Algorithms*, pp. 42–50. Morgan Kaufmann.

Farshadnia, R. (1991). *CACSD using Multi-Objective Optimization*. PhD thesis, University of Wales, Bangor, UK.

Fleming, P. J. (1985). Computer aided design of regulators using multiobjective optimization. In *Proc. 5th IFAC Workshop on Control Applications of Nonlinear Programming and Optimization*, pp. 47–52, Capri. Pergamon Press.

Fleming, P. J., Crummey, T. P., and Chipperfield, A. J. (1992). Computer assisted control system design and multiobjective optimization. In *Proc. ISA Conf. on Industrial Automation*, pp. 7.23–7.26, Montreal, Canada.

Fleming, P. J., Fonseca, C. M., and Crummey, T. P. (1993). MATLAB: Its toolboxes and open structure. In Linkens, D. A., editor, *CAD for Control Systems*, chapter 11, pp. 271–286. Marcel-Dekker.

Fourman, M. P. (1985). Compaction of symbolic layout using genetic algorithms. In Grefenstette, J. J., editor, *Proc. First Int. Conf. on Genetic Algorithms*, pp. 141–153. Lawrence Erlbaum.

Gembicki, F. W. (1974). *Vector Optimization for Control with Performance and Parameter Sensitivity Indices*. PhD thesis, Case Western Reserve University, Cleveland, Ohio, USA.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, Massachusetts.

Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J. J., editor, *Proc. Second Int. Conf. on Genetic Algorithms*, pp. 41–49. Lawrence Erlbaum.

Hancock, S. D. (1992). *Gas Turbine Engine Controller Design Using Multi-Objective Optimization Techniques*. PhD thesis, University of Wales, Bangor, UK.

MathWorks (1992a). MATLAB *Reference Guide*. The MathWorks, Inc.

MathWorks (1992b). SIMULINK *User's Guide*. The MathWorks, Inc.

Richardson, J. T., Palmer, M. R., Liepins, G., and Hilliard, M. (1989). Some guidelines for genetic algorithms with penalty functions. In Schaffer, J. D., editor, *Proc. Third Int. Conf. on Genetic Algorithms*, pp. 191–197. Morgan Kaufmann.

Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In Grefenstette, J. J., editor, *Proc. First Int. Conf. on Genetic Algorithms*, pp. 93–100. Lawrence Erlbaum.

Wienke, D., Lucasius, C., and Kateman, G. (1992). Multicriteria target vector optimization of analytical procedures using a genetic algorithm. Part I. Theory, numerical simulations and application to atomic emission spectroscopy. *Analytica Chimica Acta*, 265(2):211–225.