Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms II: Application Example

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

January 23, 1995

Research Report 565

Contents

Abstract

The evolutionary approach to multiple function optimization formulated in the first part of the paper [1] is applied to the optimization of the low-pressure spool speed governor of a Pegasus gas turbine engine. This study illustrates how a technique such as the Multiobjective Genetic Algorithm can be applied, and exemplifies how design requirements can be refined as the algorithm runs.

Several objective functions and associated goals express design concerns in direct form, i.e., as the designer would state them. While such a designer-oriented formulation is very attractive, its practical usefulness depends heavily on the ability to search and optimize cost surfaces in a class much broader than usual, as already provided to a large extent by the Genetic Algorithm (GA).

The two instances of the problem studied demonstrate the need for preference articulation in cases where many and highly competing objectives lead to a non-dominated set too large for a finite population to sample effectively. Further, it is shown that only a very small portion of the non-dominated set is of practical relevance, which further substantiates the need to supply preference information to the GA. The paper concludes with a discussion of the results.

1 Introduction

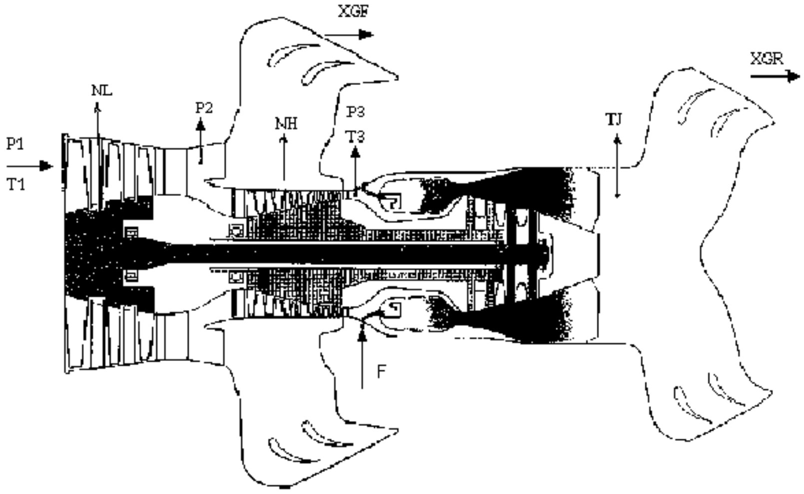
The evolutionary approach to multiple function optimization formulated in the first part of the paper [1] is applied to the optimization of the low-pressure spool speed governor of a Pegasus gas turbine engine, a non-trivial and realistic non-linear multiobjective optimization problem.

The study provides an illustration of how a technique such as the Multiobjective Genetic Algorithm can be applied. It also exemplifies how the trade-off information generated by the algorithm can contribute to a better understanding and appreciation of the complexity of the problem before requirements are refined. In particular, the concept of preferability can be seen to establish an important compromise between aggregating function approaches, which would provide a single final solution, and a pure Pareto approach, which would be faced with sampling a very large trade-off surface of which only a small portion is of practical relevance. Variable dependencies leading to ridge-shaped fitness landscapes can also be identified in the problem.

2 The gas turbine engine model

This study is based on a non-linear model of a Rolls Royce Pegasus gas turbine engine. The engine layout and the physical meaning of some of the associated variables are shown in Figure 1. The engine model reproduces dynamic characteristic variations for the fuel thrust range over the complete flight envelope.

The model of the full system also includes a fuel system model, stepper motor input and digital fan speed controller, and is shown in Figure 2 [2, 3]. The demand in low-pressure spool speed (NLDEM) is the only control input of the controller. The other inputs allow external perturbations to be applied to the system, but are not used in this work. NLDEM is compared with the actual speed of the low-pressure spool (NL) in order to produce a demand in high-pressure spool speed



F	Fuel flow
NH	High-pressure spool speed
NL	Low-pressure spool speed
P3	High-pressure delivery pressure
P2	Low-pressure delivery pressure
T3	High-pressure delivery temperature
TJ	Jet pipe temperature
XGF	Front nozzle thrust
XGR	Rear nozzle thrust

Figure 1: The Pegasus turbofan engine layout

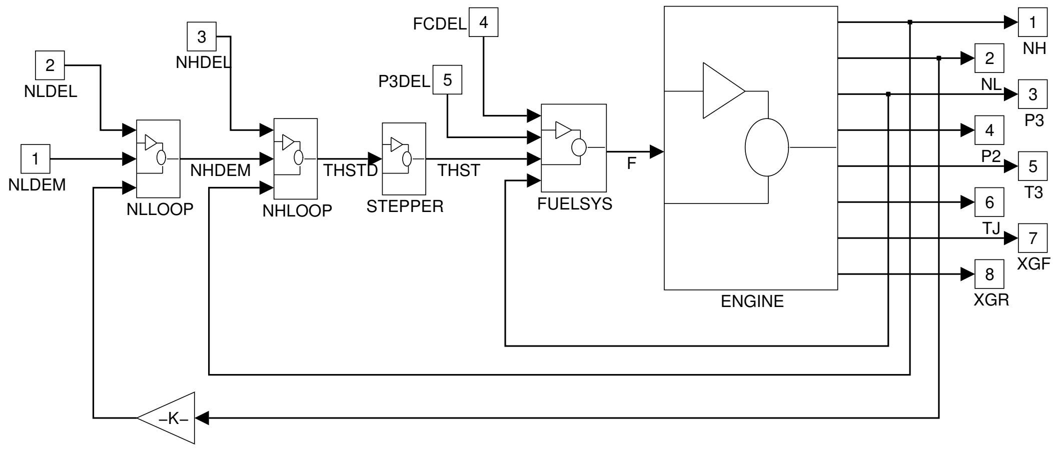


Figure 2: The SIMULINK model of the system

(NHDEM). The desired position of the stepper motor THSTD is then computed from NHDEM and the actual high-pressure spool speed (NH).

The structure of the existing controller was originally determined by Rolls Royce [4] on the basis of requirements imposed by its integration with other systems on the aircraft. Both NH and NL are states of the engine model. The remaining outputs are simply non-linear functions of the two states.

3 The design problem

Each block of the controller, NLLOOP and NHLOOP, is characterized by four parameters, one of which is a real gain value. The other three parameters are time constants. The appropriate setting of such parameters, or design variables, is cast as the problem of optimizing a number of closed-loop performance measures.

In the existing controller, the non-linearity of the plant (i.e., the stepper motor, the fuel system and the engine) is handled by scheduling different controller parameters on-line against the values of NLDEM and NH. In this study, a single operating point of the plant is considered, and the design procedure described gives but a single step towards the design of a controller for the full range of operation of the plant. Nevertheless, the example serves well to show how the concept of preferability can be effectively explored by an evolutionary technique such as the Genetic Algorithm.

3.1 Design objectives

The design objectives were extracted from the original stability and response requirements provided by the manufacturers, consisting of a selection of both time and frequency-domain objectives. Frequency-domain measures were computed from the model linearized around the operating point defined by NL = 70%. The time-domain characteristics were derived from the response of the full non-linear model to a (small) step input of 3%NL, obtained through simulation.

Closed-loop stability. The stability of the closed-loop system is probably the most basic objective to be satisfied. In the discrete-time case, the maximum pole magnitude of the corresponding closed-loop system provides a simple and effective means of assessing how far from being stabilizing a non-stabilizing controller is. If set up as a high-priority objective with unity for a goal, this measure enables stabilizing controllers to be found before other objectives come into play [5].

If, however, a fast time response is also required, the maximum pole magnitude of the final controller should continue to be minimized below unity. For this reason, this measure is set up as a soft objective here.

Gain and phase margins. The gain and phase margins are measures of robustness to multiplicative loop perturbations. They are generally independent from the maximum pole magnitude of the closed-loop system. The gain margin for the Pegasus engine is required to be at least 1.5 (3.53 dB), and the phase margin is required to be at least 50 degrees. Both margins are measured by opening the loop after the controller output.[1] The maximization of both margins is implemented as the minimization of their complements.

Rise time The time taken by the closed-loop system to reach, and stay above, 70% of the final output change demanded by a step-input. It is required to be no longer than 0.7 of the time taken by the open-loop engine to reach 70% of a similar output change, following a step in fuel flow.

Settling time The time taken by the closed-loop system to stay within $\pm 10\%$ of the final output change demanded by a step-input. It is required to be

[1] This is consistent with modelling the uncertainty associated with the fuel system in terms of maximum gain and phase deviations, although it does not directly address the behaviour of the system in the presence of that uncertainty.

no longer than the time taken by the open-loop engine to reach 90% of a similar output change, following a step in fuel flow.

Overshoot The maximum value reached by the output as a consequence of a step input should not exceed the final output change by more than 10%.

Output error Given the presence of a pure digital integrator in the control loop, the steady-state output error of the system is guaranteed to be zero. Consequently, and provided the system is fast enough, the output error measured at the end of the simulation should be close to zero. The associated goal is set to 0.1% of the desired output value.[2]

4 Implementation

The several objective functions, the multiobjective ranking algorithm, and all GA routines were implemented as MATLAB [6] M and MEX-files [7]. A graphical user interface (GUI) was also written as an M-file, making use of the graphical capabilities of MATLAB V4. The model was simulated with SIMULINK [8].

4.1 Parameter encoding

The controller parameters were encoded as 15-digit binary strings each, using Gray coding, and then concatenated to form the chromosomes. Gain parameters in the interval $[10^{-4}, 10^{3})$, and time constants in the interval $[10^{-6}, 10)$ (s), were logarithmically mapped onto the set of possible chromosome values, $\{0, \ldots, 2^{15}-1\}$, providing a resolution of about 4 significant digits in each case.

[2]This goal was taken from the steady state stability requirement, which states that speed fluctuations should not exceed $\pm 0.1\%$NL.

4.2 Genetic operators

The genetic algorithm consisted of a fairly standard generational GA with multiobjective ranking, and with sharing and mating restriction implemented in the objective domain, as described in Part I [1]. A small percentage (10%) of random immigrants was inserted in the population at each generation, with a view to making the GA more responsive to on-line preference changes [9].

After evaluation, multiobjective ranks were computed according to the current preferences, and the population sorted. Fitness assignment followed.

4.2.1 Fitness assignment and selection

The selective pressure (defined as the ratio between the number of offspring expected by the best individual in the population and the average number of offspring per individual) imposed by rank-based fitness assignment is constant and can be easily set. In particular, arbitrary values of selective pressure can be obtained by assigning fitness exponentially [10].

In a fixed-size population, the average number of offspring per individual is one and, therefore, the number of offspring expected by the best individual equals the selective pressure. In the present case, however, only 90% of the individuals in the new population are the result of selection, the remaining 10% consisting of random immigrants. Consequently, for the expected number of offspring of the best individual to be kept constant, selective pressure must increase with the percentage of random immigrants introduced in the population. This is useful in setting parameters such as the mutation rate, as discussed below.

By guaranteeing a non-zero, although small, probability of reproduction to the least fit individuals in the population for all values of selective pressure, exponential ranking also gives random immigrants a chance to take part in the next recombination stage of the algorithm. It is desired that, through recombination, they may pass their genetic diversity on to the population, since it is unlikely that randomly generated individuals are able to compete with the current best directly in terms of fitness, especially as the population evolves towards optimality.

The fitness assigned to individuals with the same multiobjective rank is averaged, and fitness shared within each rank before selection takes place. The sharing parameter σ_{share} is estimated given the number N of preferable individuals in the present generation and, for each objective, either the range they define in objective space, or the corresponding range defined by the set of preferred individuals found until, and including, the present generation, whichever is smaller. This encourages the population to expand across the trade-off surface, while allowing it to contract when some of its preferable individuals become known to lie outside the preferred region.

Selection uses Baker's stochastic universal sampling (SUS) algorithm [11], which is optimal in terms of bias and spread. It minimizes the stochastic errors associated with roulette-wheel selection, and, thus, genetic drift.

4.2.2 Recombination

Once the parents of the next generation have been selected from the old population, they are paired up and recombined with high probability (0.7). Mating restriction is implemented by forming pairs of individual within a distance σ_{mate} of each other in objective space, where possible [12]. Reduced-surrogate [13] shuffle crossover [14] is used for recombination.

4.2.3 Mutation

Although initially thought to play a secondary rôle in the context of the Genetic Algorithm [15], mutation has been found to, despite its simplicity, be an important, active search operator. Independent bit mutation is characterized by a single parameter, the bit mutation rate, or probability, the setting of which depends on the selective pressure and chromosome length [16].

If there was no crossover, the probability of an individual not actually undergoing, or surviving, mutation should be at least the inverse of the best individual's expected number of offspring μ for the best individual to, on average, be passed on to the next generation. For length ℓ chromosomes,

$$P_{\mathrm{m}} \leq 1 - \mu^{-1/\ell}$$

where P_{m} represents the bit mutation probability. In the presence of crossover, the actual P_{m} should be somewhere below the limit. For $\mu = 2$, setting

$$P_{\mathrm{m}} = 1 - (\alpha\mu)^{-1/\ell}$$

where $\alpha = 0.9$, was found to perform well. (Note that in the presence of mating restriction, crossover tends not to be too destructive).

4.3 Offspring evaluation

Only those offspring which were actually changed by the genetic operators were re-evaluated, as suggested in [17]. This simple "caching" mechanism reduced the number of objective function evaluations by between 20% and 25%.

5 Results

The setting of the gain parameters of the controller given the existing time-constant parameter values will be considered first. Involving only two decision variables, this formulation makes it possible to directly visualize cost surfaces in the decision variable domain.

The optimization of all eight controller parameters follows, with emphasis being put on the articulation of preferences in the objective domain. The seven objectives described above are included in both cases.

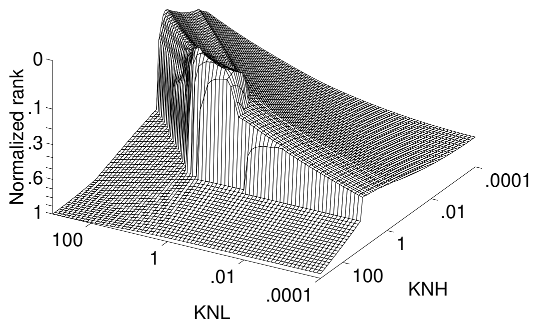


Figure 3: The cost surface associated with the maximum pole magnitude of the closed-loop system.

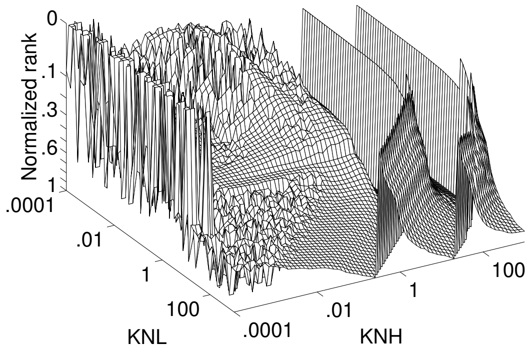


Figure 4: The cost surface associated with the phase margin of the closed-loop system.

5.1 A two-variable example

Consider the problem of minimizing the seven objectives by varying only the controller gains, KNL and KNH. Since there are only two variables, the cost surfaces associated with each of the objectives can be produced through gridding and visualized, although this is a computationally intensive task.

Figures 3 to 6 depict four of the seven single-objective cost surfaces involved in this problem. Due to the bad scaling of some of these objectives, the points on the grid were ranked after evaluation, as discussed in Part I [1]. Whilst being closer to how a GA with ranking would see them, the resulting surfaces are much easier to visualize than the original ones. In each case, the axes have been rotated so as to highlight the features of each surface.

It is possible to see how some objectives, notably those defined in the time domain, tend to display large flat regions corresponding either to lowest performance, as with settling time (Figure 5) and rise-time (not shown), or even to optimal performance in that objective's sense, as with overshoot (Figure 6). However, zero (optimal) overshoot means very little on its own: for example, it may imply too long a rise-time.

In these regions of the search space, the optimizer must rely on other objec-

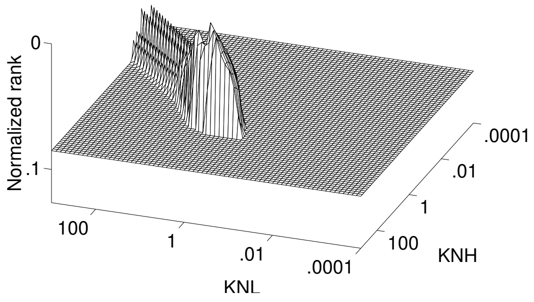


Figure 5: The cost surface associated with the settling time of the closed-loop step response.

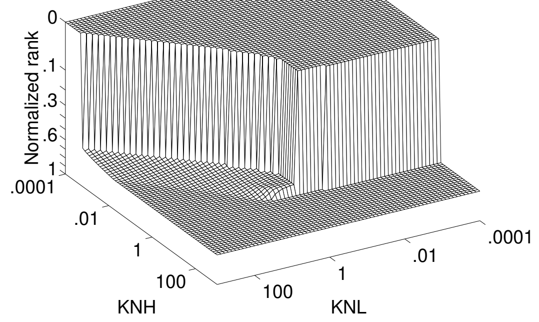


Figure 6: The cost surface associated with the overshoot of the closed-loop step response.

tives, such as the maximum closed-loop pole magnitude (Figure 3). The closed forms used to compute the gain and phase margins, though resulting in fairly fast numerical algorithms, proved not to be sufficiently numerically robust across the whole search space. This is apparent in Figure 4 for the phase margin. Nevertheless, the algorithms did produce accurate results with "sensible" gain settings, and were used instead of more stable, but brute-force, approaches. (A discussion of genetic search with approximate function evaluations can be found in [18]).

One can also estimate and plot a representation of the cost surface associated with the Pareto formulation of the problem when no goals are specified (Figure 7). Here, the most striking features are, perhaps, the large area of the search space corresponding to non-dominated solutions and the irregular shape of that region, also shown in Figure 8.

Specifying goals for the objectives defines a much smaller area of preferable solutions (Figures 9 and 10), whose dimensions approach the resolution of the grid. As it turns out, none of these points is, in fact, satisficing. Actual preferred solutions, whether satisficing or not, will probably be confined to an even smaller region of the search space.

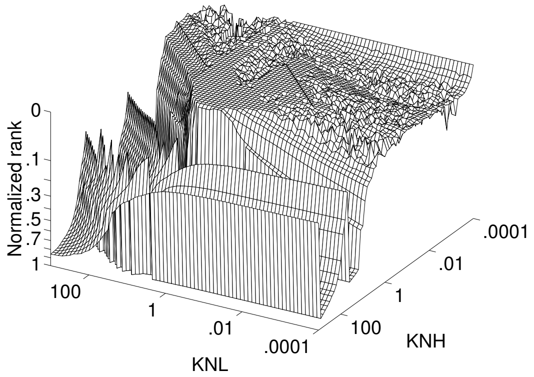


Figure 7: A representation of the cost surface associated with the Pareto-based formulation in the absence of preference information.

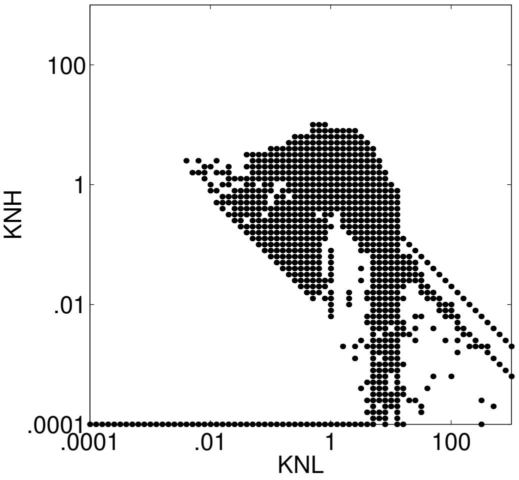


Figure 8: An alternative view of the non-dominated set estimated through gridding.

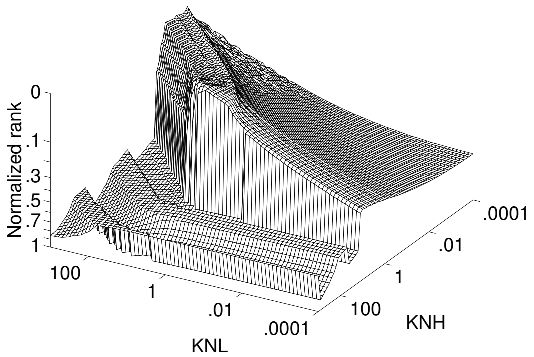


Figure 9: A representation of the cost surface associated with the formulation including goal information.

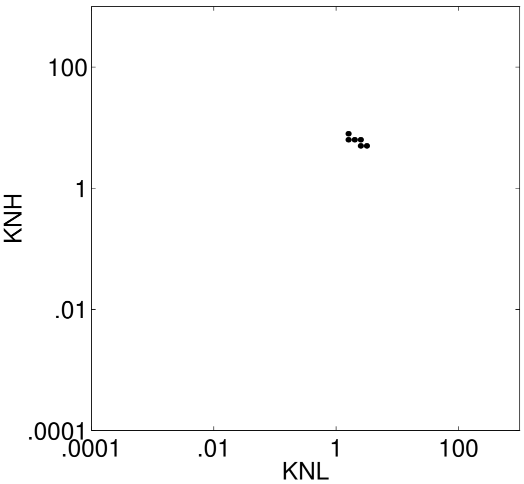


Figure 10: An alternative view of the preferable set estimated through gridding.

5.1.1 Genetic algorithm results

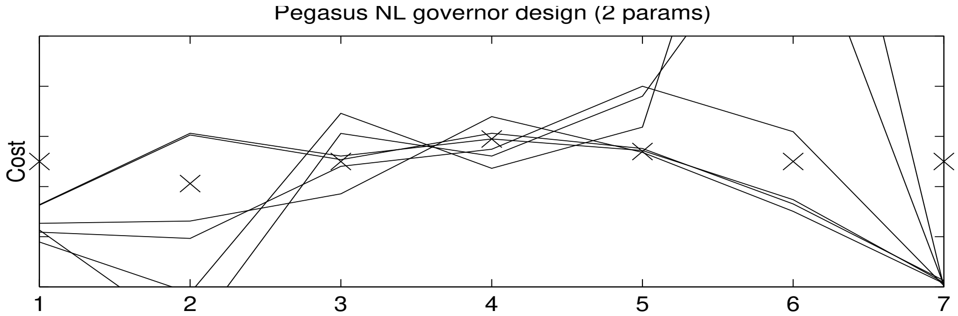
A genetic algorithm with a population size of 100 individuals was run on this problem five times, each time for 50 generations. Out of a total of about 4000 points evaluated in each run, around a half were non-dominated relatively to each other. Of these, from 10 to 200 points were satisficing. Runs in which the population could locate itself earlier on the satisficing region rapidly produced variations of the first satisficing individuals found, which explains why some runs were apparently much more successful than others.

In this example, some degree of preference information proves to be absolutely necessary to guide the GA through a very large non-dominated set towards solutions of practical interest. Figure 11 illustrates the solutions so-far-preferable at three different stages of an algorithm run. The preferable solutions found at the end of the run reveal trade-offs between the several objectives within the bounds imposed by the goals associated with them.

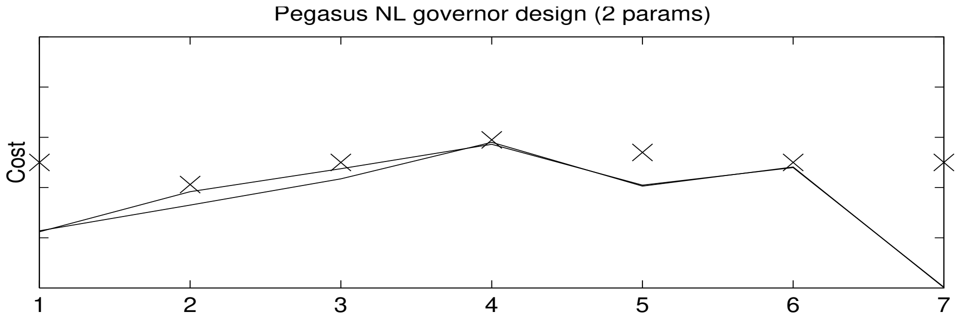
In these plots, objectives are represented by their integer index i on the x-axis, and performance in each objective dimension is plotted on the y-axis. Each candidate solution $\mathbf{x}$ is represented by a line connecting the points $(i, f_i(\mathbf{x}))$. A trade-off between adjacent objectives results in the crossing of lines between them, whereas non-concurrent lines indicate that those objectives are non-competing. For example, rise-time (objective 4) and settling-time (objective 5) do not seem to compete heavily. In turn, there is a clear trade-off between these two objectives and objective 6 (overshoot). The gain and phase margins (objectives 2 and 3) seem to be competing objectives in part of the trade-off only.

5.2 Full controller parameter optimization

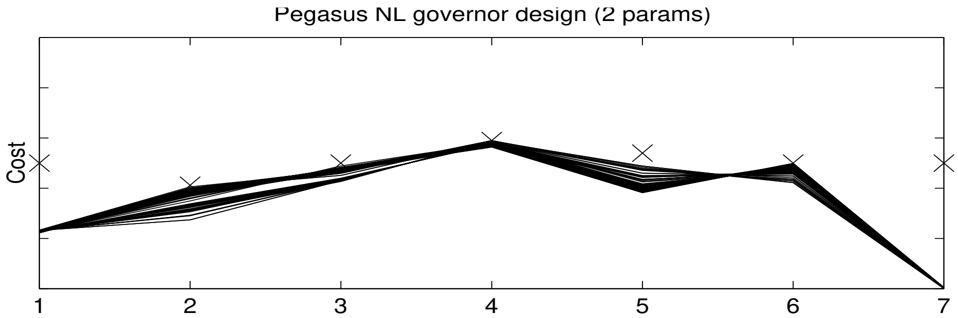
The optimization of the full controller assumes no a priori knowledge of the controller parameter values apart from that implicitly used in setting the parameter ranges. Individuals are now 120-bit long, and the population is increased to 150



After 10 generations



After 30 generations



After 50 generations

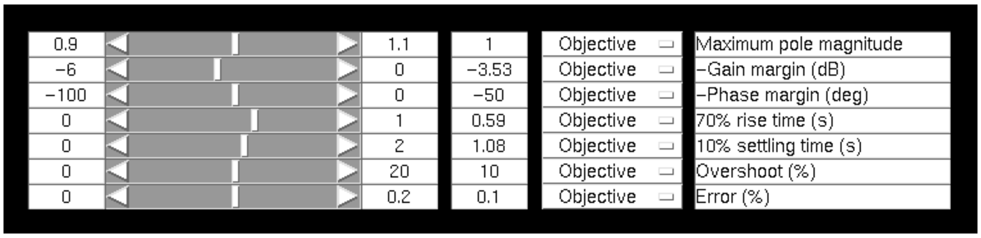


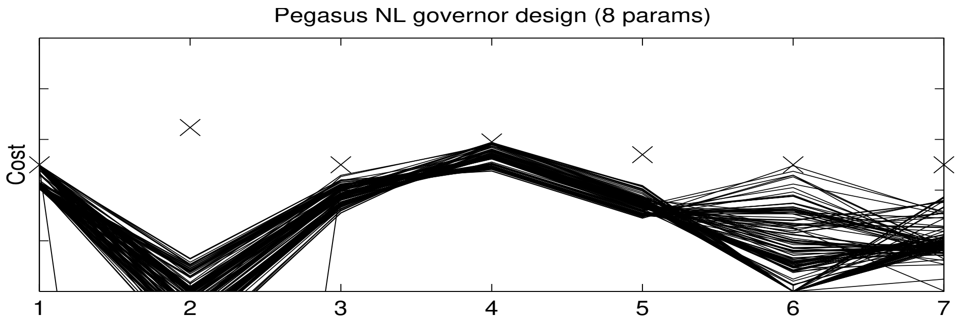
Figure 11: Sample trade-off graphs from a run on the 2-variable example, showing the preferable solutions found after 10, 30 and 50 generations. The vertical axis ranges are the same as those associated with the control-panel sliders. The goal values are given in the middle column of the panel and are plotted as “$\times$”.

individuals.

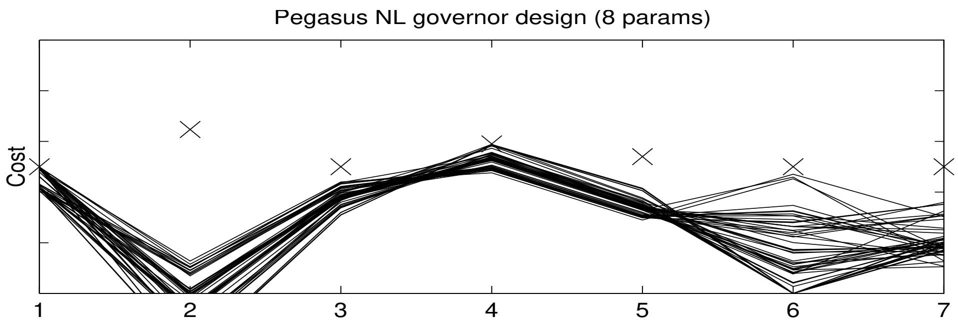
Running the GA reveals a much larger set of satisficing solutions. The preferable individuals found until, and including, the 40th generation are shown in Figure 12(a). When compared with the previous example, the solutions found have all greater maximum pole-magnitudes, that is, longer time-constants. On the other hand, they generally exhibit better stability margins and settling time. Thus, not only do the solutions found in this second example not dominate those found previously, but also they represent a significantly different region of the preferable set.

The fact that the 8-variable GA could not find solutions similar to, or better than, those found for the 2-variable example shows how sampling the whole preferred region with a population of a given size may not be always feasible, even though niche induction mechanisms are used. The diversity among the preferable individuals in the population at generation 40, shown by the corresponding trade-off graph in Figure 12(b), is an indication that these techniques are indeed at work, at least to some extent. Nevertheless, their effectiveness in causing the population to spread across the whole, actual, preferred set also depends on the genetic algorithm's own ability to capture the regularities in the fitness landscape, and generate offspring accordingly. Unfortunately, as discussed in Part I [1], binary coded GAs cannot be expected to be able to deal well with strong variable dependencies.

Appropriate interaction with the decision maker can help address this difficulty. The algorithm has found solutions which do, in principle, meet their goals, and which it sees all as equally fit. Other solutions non-preferable to these may exist, but they are probably more difficult to find given the encoding structure used for the chromosomes and the associated genetic operators. If the DM finds the candidate solutions produced by the GA unacceptable, then preferences should be refined so as to stimulate the GA to move on to a different region of the non-dominated set.



(a) Cumulative trade-off graph after 40 generations



(b) Preferable individuals in the population at generation 40

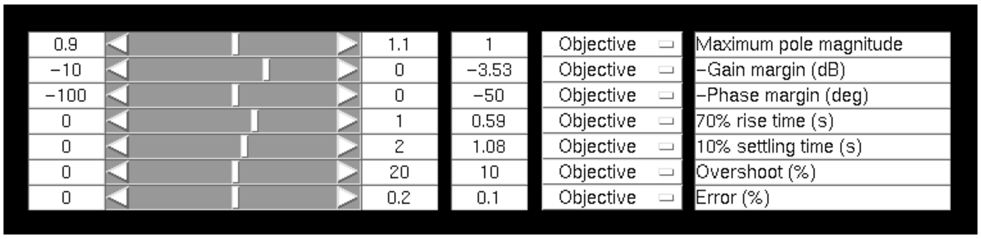
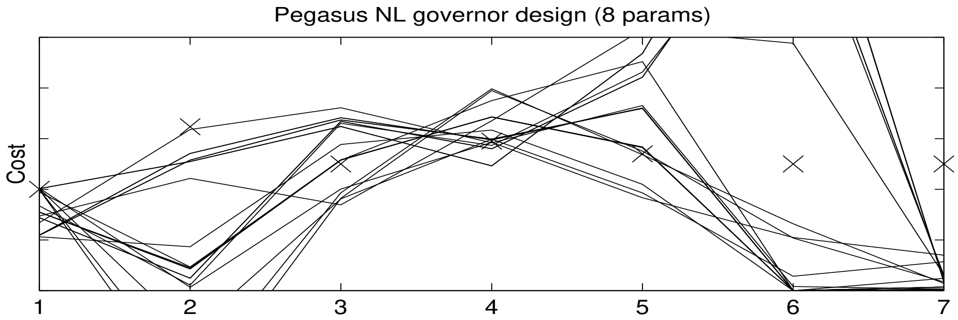


Figure 12: Sample trade-off graphs from a run on the 8-variable example, showing preferable solutions found after 40 generations. The vertical axis ranges are the same as those associated with the control-panel sliders. The goal values are given in the middle column of the panel and are plotted as a "$\times$".

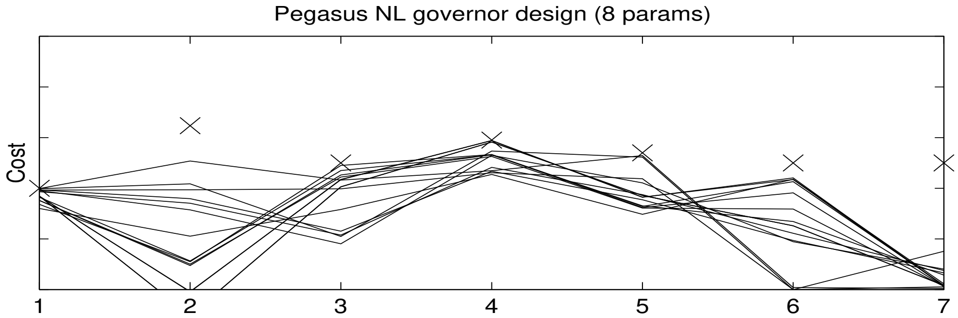
As an example, consider again the trade-off graph in Figure 12(a). The step-response error at the end of the simulations is not particularly low, reflecting the long time constants associated with maximum pole magnitudes close to unity. If faster responses are desired, the DM can tighten the goals associated with either, or both, of these two objectives while continuing to run the algorithm.

Suppose the DM chooses to lower the goal associated with the maximum pole magnitude to 0.98. Solutions capable of meeting this new requirement turn out not to have yet been found, and a set of relatively non-dominated solutions close to being satisficing is presented to the DM (Figure 13(a)). If the DM *believes* this new set of goals can in fact be met (for example, because some goals in the previous set have clearly been over-attained), there is no algorithmic reason not to specify it. If the GA cannot find satisficing solutions, it will at least produce a set of relatively non-dominated approximations to the new goals on which further preference changes can be based. As the population continues to evolve, reductions in maximum pole magnitude are accompanied by reductions in the step-response error measured at the end of the simulation, and satisficing solutions are eventually found at the expense of some degradation in gain margin and other objectives (Figure 13(b)).

Further refinements could involve specifying hard, instead of soft, bounds for some of the objectives. Suppose the DM decided that a gain margin of 6dB seemed possible (in the presence of the trade-off graph), and that better gain and phase margins would not be required. Raising the priority of these objectives, and changing the gain margin goal, converts them into hard constraints as desired. Doing so reduces the preferred set further (Figure 14(a)), and the process continues. The family of step responses corresponding to the trade-off graph in Figure 14(b) is shown in Figure 15.



(a) After 40 generations (new preferences)



(b) After 60 generations

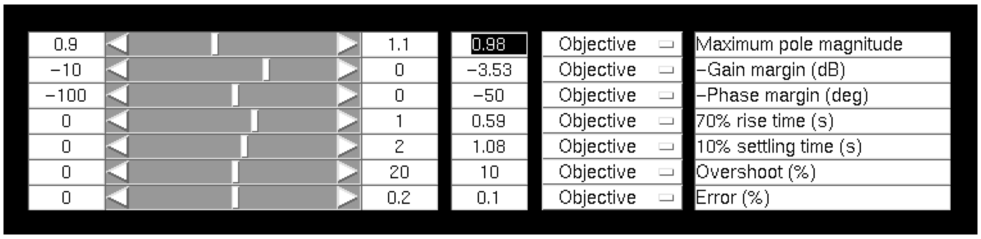
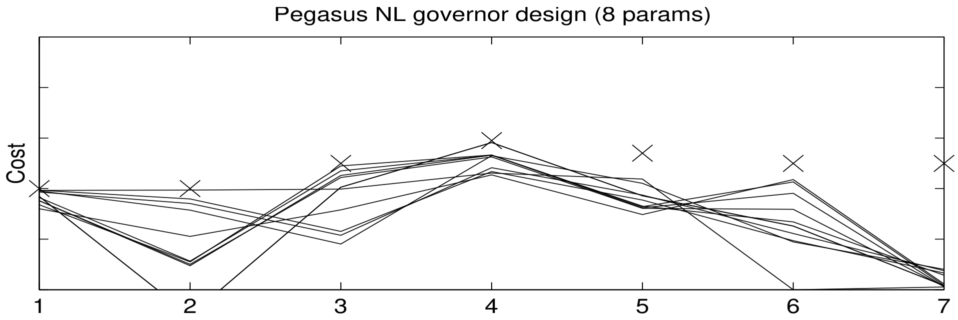
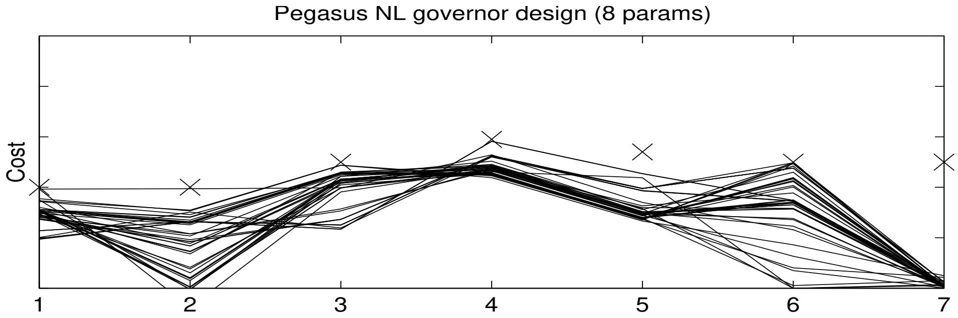


Figure 13: Sample trade-off graphs showing the effect of tightening one goal's value. The population is driven towards achieving that goal at the expense of degradation with respect to others.



(a) After 60 generations (hard stability margins)



(b) After 80 generations

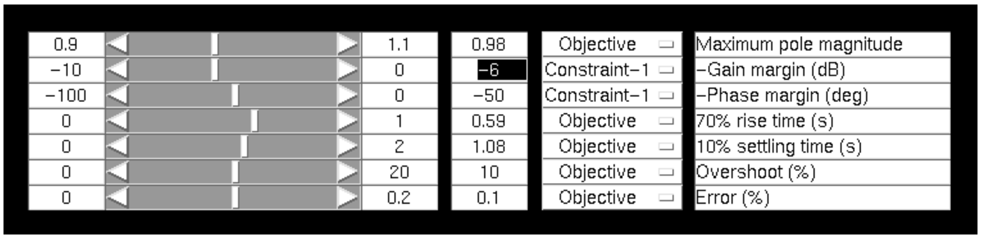


Figure 14: Sample trade-off graphs showing the effect of converting two objectives into hard constraints, while also changing the goal associated with of one of them.

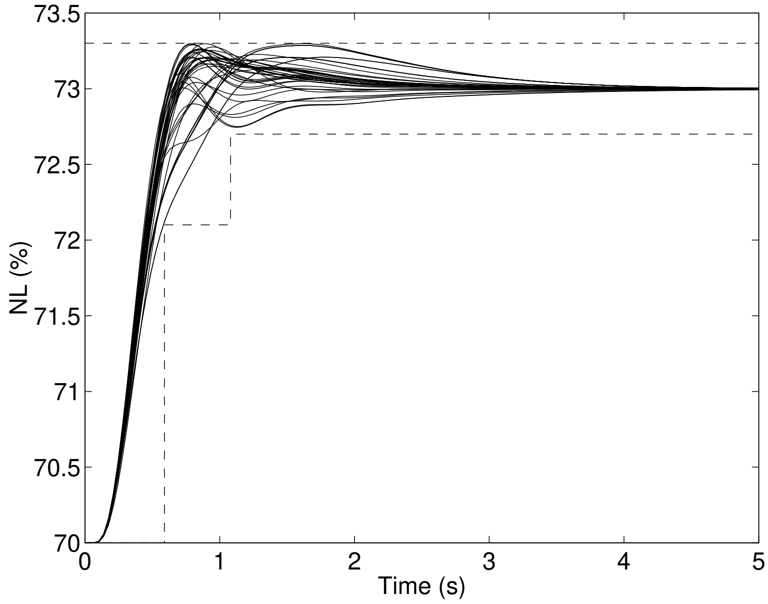


Figure 15: Family of preferable step-responses found after 80 generations.

6 Discussion

In the authors' search for a reduced, yet illustrative, set of realistic design objectives for the Pegasus engine, the ability of the GA to exploit omissions in the design requirements soon became apparent. The set up of the closed-loop maximum pole magnitude as a soft objective instead of a hard one was a direct consequence of the solutions produced by early runs of the algorithm being clearly unacceptable: the GA was sometimes able to exploit the non-linearity of the plant in such a way that the step-response characteristics achieved were only valid for the particular amplitude of the step used during simulation. Applying slightly larger or smaller steps to the controlled plant immediately revealed the slow dynamics in the closed loop, deeming the controller unacceptable.

The objective functions and respective goals used in these examples reflect design concerns in a direct form. The problems were formulated from the point of view of the designer, not that of the optimizer. The associated cost landscapes do not all meet common requirements such as continuity, smoothness, and absence of flat regions, and neither does the resulting multiobjective cost landscape. The practical usefulness of such a formulation is, therefore, tied to the ability to search and optimize cost surfaces in a class much broader than usual, as provided by the GA.

On the other hand, GAs do work best when no strong dependencies between decision variables exist, which was neither the case for the simple example given in Part I [1], nor is the case for the Pegasus controller design example. Evolutionary algorithms to be used in multiobjective parameter optimization should undoubtedly address this issue, because it seems to arise from the very nature of the Pareto formulation. Decision variable dependencies have been taken into account mainly in the context of Evolution Strategies [19], at the coding level. Simple mating restriction techniques, as described and discussed in Part I [1], may require too large a population to become effective as the size and complexity of the trade-off surfaces increase, while also remaining practical. The preferability relation provides a means of addressing this difficulty by interfacing between the DM and the search algorithm, and allowing the two to interact.

When objective functions are very expensive to evaluate, the interactive use of the MOGA may become less attractive due to the long time spent at the function evaluation stage of the GA. In such generational EAs, however, many candidate solutions can be evaluated independently from each other, making possible the use of parallelism paradigms which exploit event replication, while largely avoiding issues such as load balancing and inter-processor communication problems. Parallel formulations (as opposed to parallel implementation) of EAs are also important alternatives, because of the niche mechanisms which arise in geographically isolated populations.

Finally, evolutionary algorithms are not restricted to parameter optimization. They are also, and probably mainly, powerful combinatoric optimizers. The preferability relation can, in principle, be applied to virtually all design applications of evolutionary algorithms, whether they involve permutation problems such as scheduling and the travelling salesman problem [20], grouping problems such as process-processor mapping [21] and non-linear model term selection [22], or the evolution of computer programs [23].

7 Concluding remarks

The application of the multiobjective genetic algorithm to a controller design example was used to demonstrate the need for some degree of preference articulation in Pareto-based evolutionary optimization. The preferability relation provides such a mechanism, allowing the designer to concentrate on design requirements rather than on the properties of the final cost function to be optimized.

The interactive use of such an evolutionary multiobjective algorithm is very attractive. Speed of execution then becomes an important concern, which can possibly be addressed with parallel processing techniques. The need for improved EAs capable of addressing strong variable dependencies to some extent has also been highlighted.

Acknowledgement

Thanks are due to A. Shutler of Rolls-Royce for sharing his expertise and knowledge in connection with the Pegasus case-study. The first author gratefully acknowledges support by Programa CIENCIA, Junta Nacional de Investigação Científica e Tecnológica, Portugal. The authors also wish to acknowledge the support of the UK Engineering and Physical Sciences Research Council (Grant GR/J70857) in completion of this work.

References

[1] 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, Sheffield, U.K., Jan. 1995.

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

[3] "Turbofan engine model," technical report, Control and CAD Group, University College of Swansea, Swansea, U.K., June 1991. Final draft.

[4] A. G. Shutler, "A case study problem for the advanced controls technology club," Internal Report CSAN 2790, Rolls Royce plc, Bristol, U.K., 1991.

[5] C. M. Fonseca and P. J. Fleming, "Multiobjective optimal controller design with genetic algorithms," in *Proc. IEE Control'94 International Conference*, vol. 1, (Warwick, U.K.), pp. 745–749, 1994.

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

[7] 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, Sheffield, U.K., July 1994.

[8] The MathWorks, Inc., SIMULINK *User's Guide*, Mar. 1992.

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

[10] P. J. B. Hancock, "An empirical comparison of selection methods in evolutionary algorithms," in *Evolutionary Computing, AISB Workshop* (T. E. Fogarty, ed.), vol. 865 of *Lecture Notes in Computer Science*, pp. 80–94, Berlin: Springer-Verlag, 1994.

[11] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (J. J. Grefenstette, ed.), pp. 14–21, Lawrence Erlbaum, 1987.

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

[13] L. Booker, "Improving search in genetic algorithms," in *Genetic Algorithms and Simulated Annealing* (L. Davis, ed.), Research Notes in Artificial Intelligence, ch. 5, pp. 61–73, London: Pitman, 1987.

[14] R. A. Caruana, L. J. Eshelman, and J. D. Schaffer, "Representation and hidden bias II: Eliminating defining length bias in genetic search via shuffle crossover," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (N. S. Sridharan, ed.), pp. 750–755, Morgan Kaufmann, 1989.

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

[16] I. Harvey, "Evolutionary robotics and saga: The case for hill crawling and tournament selection," Cognitive Science Research Paper 222, University of Sussex, Brighton, Brighton, U.K., 1992.

[17] P. Oliveira, J. Sequeira, and J. Sentieiro, "Selection of controller parameters using genetic algorithms," in *Engineering Systems with Intelligence* (S. G. Tzafestas, ed.), pp. 431–438, the Netherlands: Kluwer Academic, 1991.

[18] J. J. Grefenstette and J. M. Fitzpatrick, "Genetic search with approximate function evaluations," in *Genetic Algorithms and Their Applications: Pro-*

ceedings of the First International Conference on Genetic Algorithms (J. J. Grefenstette, ed.), pp. 112–120, Lawrence Erlbaum, 1985.

[19] T. Bäck, F. Hoffmeister, and H.-P. Schwefel, "A survey of evolution strategies," in *Genetic Algorithms: Proceedings of the Fourth International Conference* (R. K. Belew and L. B. Booker, eds.), pp. 2–9, San Mateo, CA: Morgan Kaufmann, 1991.

[20] D. Whitley, T. Starkweather, and D. Shaner, "The travelling salesman and sequence scheduling: Quality solutions using genetic edge recombination," in *Handbook of Genetic Algorithms* (L. Davis, ed.), ch. 22, pp. 350–372, New York: van Nostrand Reinhold, 1991.

[21] M. J. Baxter, M. O. Tokhi, and P. J. Fleming, "Heterogeneous architectures for real-time control: Design tools and scheduling issues," in *Preprints of the 12th IFAC Workshop on Distributed Computer Control Systems (DCCS'94)* (J. A. del al Puente and M. G. Rodd, eds.), (Toledo, Spain), pp. 89–94, 1994.

[22] C. M. Fonseca, E. M. Mendes, P. J. Fleming, and S. A. Billings, "Non-linear model term selection with genetic algorithms," in *IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, vol. 2, (Essex, U.K.), pp. 27/1–27/8, 1993.

[23] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, Massachusetts: MIT Press, 1992.