A real application example of a control structure selection by means of a multiobjective genetic algorithm

M. Parrilla Sánchez and J. Aranda Almansa

Departamento de Informática y Automática. Facultad de Ciencias. Universidad Nacional de Educación a Distancia (UNED). 28850 Madrid. España,
{jaranda, mparrilla}@dia.uned.es

Abstract. Control problems are clear examples of multiobjective optimization. In this kind of problems a series of objectives, some of them opposed to each other, will be optimized in order to fit some design specifications.
Moreover, evolutionary algorithms have been shown to be ideal for the resolution of these kinds of problems because they work simultaneously with a set of possible solutions, thereby favoring convergence towards a global optimum. In this document we propose a way of dealing with the different objectives considered and a genetic-evolutionary algorithm that will enable some phases of the controller design to be automated.
Finally, an application example of the methods outlined will be applied to the design of a controller to reduce the sickness index on a high-speed ship.

1 Introduction

Many problems in science and engineering require the simultaneous optimization of multiple objective functions. It will therefore be necessary to optimize a function of the form $f : S \to T$ where $S \subset \Re^n$ and $T \subset \Re^m$. However, the problem is that there is not usually an element in S producing an optimum simultaneously for each of the m objective functions. This is due to a conflict among objectives. Thus, improvement in one of them gives rise to deterioration in another. Consequently, it will be necessary to reach a compromise or trade-off solution in which all the objectives are satisfied in an acceptable degree from the point of view of the design.

Evolutionary algorithms have demonstrated good behavior in the solution of multiobjective optimization problems. Evolution in these kinds of algorithms is achieved by evaluating each solution coded in the chromosome population in order to see its fitness and compare the results of this evaluation so that the most suitable solutions have a higher probability of reproducing and transmitting their characteristics to their offspring.

When comparing fitness a new problem arises. What should be compared in the treatment of multiple objectives satisfied to a greater or lesser extent by the different chromosomes of the population?

Some ways of approaching this last problem [4] will be commented on below where a new method is proposed. Lastly, this method will be applied to a specific problem.

J. Mira (Ed.): IWANN 2003, LNCS 2687, pp. 369-376, 2003.

1.1 Methods based on the notion of Pareto dominance

These methods are based on the notion of Pareto dominance. Among them, those used by Fonseca and Fleming [5] or by Srinivas and Deb [8] can be mentioned.

1.2 Methods not based on Pareto dominance

Other approaches not based on the notion of Pareto dominance have been used by other authors [4], such as the weighted sum, lexicographic ordering, or the goal programming methods [3] and [6].

2 Method of variable priorities for multiobjective evolutionary optimization

The method proposed below is inspired by lexicographic ordering and the goal programming methods, and the notion of Pareto dominance. It has been used successfully by the authors in earlier works [1] and [2].

Given $S \subset \Re^n$ and $T \subset \Re^m$, the function to optimize will be of the form $f : S \to T$. Another function $g : S \to T$ will be taken as the evaluation function of the evolutionary algorithm, where the m values returned by g are obtained by normalization of the m values returned by f.

2.1 Normalization

To carry out normalization, a vector will be established with the desired goals for the different objectives (in control problems, they will be based on the design specifications); let $o = \{o_1, ..., o_m\} \in T$ be this vector. For a chromosome x, the normalized values will be:

$$g_i(x) = \frac{o_i - f_i(x)}{o_i} \tag{1}$$

By means of normalization every objective becomes comparable to another. The original problem turns into a minimization problem.

2.2 Ranking

Population ranking is done in a similar way to the lexicographic ordering method [4], i.e., comparing the fitness vectors of the chromosomes bearing in mind the priorities of the objectives. The difference is that in the current method, priorities are not fixed, and even chromosomes in the same population will have different priorities.

Let $u = g(x_u)$ and $v = g(x_v)$ be the fitness vectors returned by the evaluation function for two different chromosomes and let $\gamma(u)$ and $\gamma(v)$ be the respective permutations of their elements, arranging them in decreasing order. $\gamma_i(u)$ and $\gamma_i(v)$ will denote their *i*th elements.

Consider also the disjointed sets *C* and *D*, whose elements will be indexes of the different objective functions, created in the following way:

$$\forall i \in \{1,\ldots,m\} \begin{cases} i \in C \Leftrightarrow \gamma_i(u) < \gamma_i(v) \\ i \in D \Leftrightarrow \gamma_i(u) > \gamma_i(v) \end{cases} \quad (2)$$

Definition 1 (preferability) Vector *u* is *preferable* to vector *v* ($u \succ v$), if and only if $C \neq \phi$ and also:

$$D = \phi \quad \text{or} \quad \min(C) < \min(D) \quad (3)$$

Definition 2 (equivalence) Vector *u* is *equivalent* to vector *v* if and only if $C = D = \phi$.

The following theorems, whose demonstrations are trivial, fulfill the requirements to carry out population ranking, in function of their fitness vectors:

Theorem 1. If vector *u* is not preferable to vector *v*, and both vectors are not equivalent, then vector *v* is *preferable* to vector *u*.

Theorem 2 (transitive property) If $u \succ v$ and $v \succ w$ then $u \succ w$.

The idea is that the objective farthest from its goal will have the greatest priority in each chromosome. Population ranking will be made according to fitness vector preferability.

3 A multiobjective genetic-evolutionary algorithm for selecting and tuning controllers

Evolutionary algorithms have been used successfully to solve multiple problems, among them control problems.

The algorithm presented here is one step further in the process of automation of controller design. A genetic algorithm will try to determine the best controller structure to use.

Two loops can be distinguished inside the algorithm. One is external, consisting of a genetic algorithm with chromosomes made up of binary numbers and using the mutation and crossover operators in an attempt to determine the best control structure. The other is internal, consisting of an evolutionary algorithm whose chromosomes will be made up of real numbers and using operators adapted to this kind of codification (see [7]) that will take charge of evaluating each one of the controller structures obtained by the external algorithm.

3.1 Internal loop

Figure 1 shows a typical control system diagram. The controller block will do the plant to track the input to the system, called a reference signal. Once the structure of the controller block has been determined, the tuning process will determine the values of a series of parameters for this block.

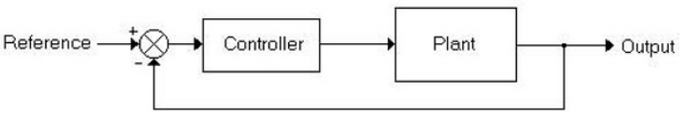

Fig. 1. Control diagram

The tuning process was carried out by the authors in [1] and [2] by means of an evolutionary algorithm, using in its fitness evaluation function the multiobjective optimization method explained in Section 2. This evolutionary algorithm will constitute the internal loop of the most general algorithm that is being presented.

Chromosomes will be coded as real number vectors, with as many elements as controller parameters need to be determined. The tournament selection technique and usual mutation and crossover operators for real number chromosomes will be used. The evolutionary algorithm will have the following steps:

1. The initial population is chosen randomly.
2. Chromosomes are decoded and evaluated by means of a computer-experiment simulation. The population is sorted according to fitness.
3. If the end conditions are given, the program concludes.
4. Tournament selection is used.
5. A new population is obtained from the selected chromosomes by means of mutation and crossover operators. In order to favor diversity and avoid premature convergence, a small number of immigrants are added (chromosomes obtained randomly).
6. Return to step 2.

This process will be repeated for each one of the controller structures determined by the external loop.

3.2 External loop

The external loop algorithm is built as a new layer or level over the internal loop just described. The objective of the external loop algorithm is to determine, by means of genetic techniques, the best control structures.

The controller will be formed by series connection with some of the following basic control subblocks:

1. Gain: K.
2. Simple pole and zero: $\frac{p}{s+p}$ y $\frac{s+p}{p}$

3. Simple lead or lag: $\frac{s+a}{s+b}$
4. Second order pole or zero: $\frac{\omega^2}{s^2+2\xi\omega s+\omega^2}$ y $\frac{s^2+2\xi\omega s+\omega^2}{\omega^2}$
5. Notch: $\frac{s^2+2\xi_1\omega s+\omega^2}{s^2+2\xi_2\omega s+\omega^2}$

Chromosomes will be made up of as many genes as different basic subblocks and each gene will be associated to one of these subblocks. Genes will take binary values, so that a 1 will indicate the presence of the corresponding subblock in the controller and a 0 will indicate its absence.

Every structure built from basic subblocks -external loop chromosomes- will be passed to the internal loop for evaluation. The internal loop will build the controller by decoding the external loop chromosome and will obtain the best values for the controller parameters. In the external loop algorithm:

1. The initial population is randomly obtained.
2. Chromosomes are evaluated by the internal loop. Population is arranged according to fitness.
3. If the end conditions are given, the program concludes.
4. Tournament selection is used.
5. A new population is obtained from the selected chromosomes by applying mutation and binary crossover operators. In order to favor diversity and avoid premature convergence, a small number of immigrants is added (chromosomes obtained randomly).
6. Return to step 2.

In short, the external loop will select the controller, while the internal loop will tune it.

4 Application example

In the CRIBAV project (Robust and Intelligent Control for High-Speed Crafts), the aim is to make robust controllers acting on some control surfaces (Flap and T-foil, see detail in Figure 2) absorb pitch and heave movements of a high-speed passenger ship. Additional information can be found on the Web: http://ctb.dia.uned.es/cribav/.

The aim will be to reduce passenger sickness index. In order to achieve this, it will be necessary to reduce the vertical accelerations of the ship due to waves. The following expression will be used for vertical acceleration, measured 40 meters away from the gravity center of the ship:

$$acv40(t_i) = a_{VH}(t_i) + a_{VP}(t_i) = \frac{d^2 heave(t_i)}{dt^2} - 40 \cdot \frac{\pi}{180} \cdot \frac{d^2 pitch(t_i)}{dt^2} \quad \textbf{(1)}$$

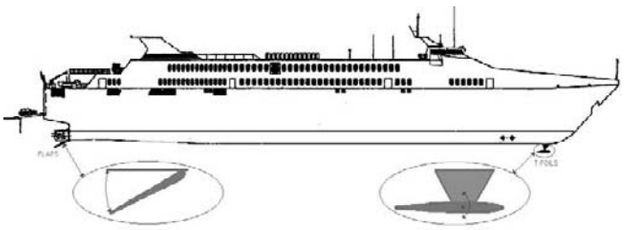

Fig. 2. Control surface detail

From the previous expression, the mean acceleration will be calculated for an experiment (by means of computer simulation). Considering N sampling instants, the expression for the mean acceleration will be:

$$J = \frac{1}{N}\sum_{i=1}^{N} \left| acv40(t_i) \right| \tag{2}$$

This expression will be employed by the evaluation function of the evolutionary algorithm to obtain a suitability measurement for a specific controller.

Different objectives considered in the evaluation function were:

- Stability of the system Plant + Controller.
- Saturations in the Flap: $0º \leq value \leq 15º$.
- Saturations in the Tfoil: $-15º \leq value \leq 15º$.
- Mean acceleration: *J*.

Because of the high computational cost required, the algorithm was implemented in C and parallelized using the specialized library MPI. The algorithm is easily parallelizable, bearing in mind that each chromosome of the external loop population can be evaluated by a different processor, and the results obtained can then be grouped. The program was executed on a Silicon Graphics Origin 2000 computer, using 30 processors, one for each external loop chromosome.

The program, once it had been designed, was executed for three different speed and sea state conditions. The sickness index reduction of the ship without control appears in the following table for different speed and sea state conditions:

Speed (Knots)	Sea State	Sickness index reduction
20	4	12,7%
40	4	46,4%
40	5	13,6%

The best results were obtained for speed = 40 knots and sea state 4. For these conditions, the controller obtained is shown in Figure 3, and the acceleration evolution and the MSI (Motion Sickness Incidence) for a range of encounter frequencies, with and without control, are shown in Figure 4.

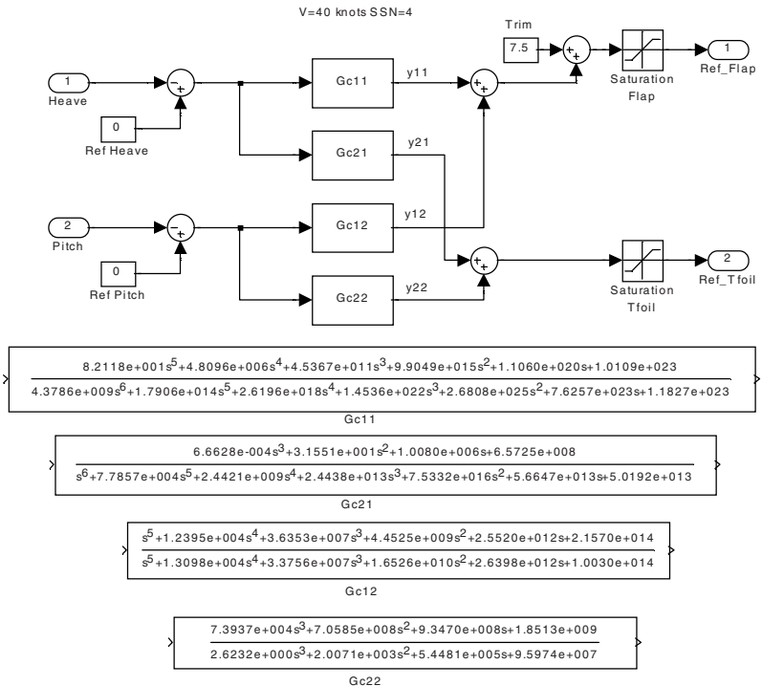

Fig. 3. Controller obtained for V = 40 knots and sea state 4.

Acknowledgment

Part of this development was supported by MCyT of Spain under contract DPI2000-0386-C03-01, and the publication is supported by a special action of the "Research Promotion Plan of UNED".

The authors also would like to thank Rafael López from the *Centro de Supercomputación de la Universidad Complutense de Madrid* for his support and the use of the university's facilities.

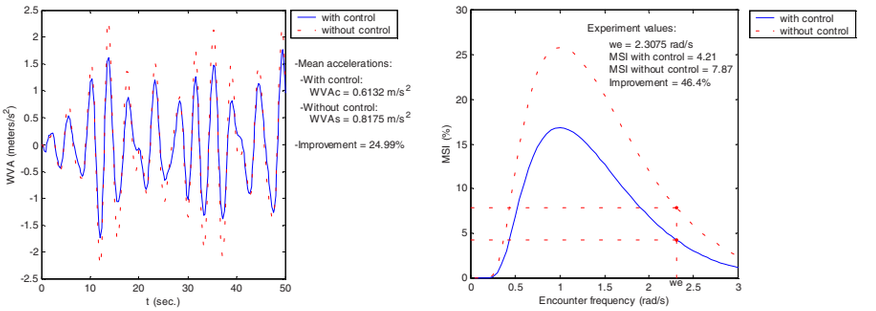

Fig. 4. Evolution of the acceleration and MSI for a range of encounter frequencies

References

1. Aranda, J.; de la Cruz, J. M.; Parrilla, M., and Ruipérez, P.: *Design of a Linear Quadratic Optimal Control for Aircraft Flight Control by Genetic Algorithm.* Controlo'2000: 4th Portuguese Conference on Automatic Control. Guimarães-Portugal. October-2000. Conference Proceedings.
2. Aranda, J.; de la Cruz, J. M.; Parrilla, M., and Ruipérez, P.: *Evolutionary Algorithms for the Design of a Multivariable Control for an Aircraft Flight Control*. AIAA Guidance, Navigation and Control Conference and Exhibit. Denver-Colorado, USA. August 2000. Conference Proceedings.
3. Charnes, A., and Cooper, W. W.: *Management Models and Industrial Applications of Linear Programming*, volume 1. John Wiley, New York, 1961.
4. Coello Coello, Carlos A.: *A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Tecniques*. Knowledge and Information Systems. An International Journal, 1(3): 269-308, August 1999.
5. Fonseca, Carlos M., and Fleming, Peter J.: *Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms-Part I: A Unified Formulation*. IEEE Transactions on Systems. Man and Cybernetics. Part A: Systems and Humans. Vol. 28. No. 1, January 1998.
6. Ijiri, Y.: *Management Goals and Accounting for Control*. North-Holland, Amsterdam, 1965.
7. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag Berlin Heidelberg. 1994.
8. Srinivas, N., and Deb, K.: *Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms*. Journal of Evolutionary Computation, Vol. 2, No. 3, pages 221-248. 1994.