COBEM 99
XV Congresso Brasileiro de Engenharia Mecânica
15th Brazilian Congress of Mechanical Engineering
22 - 26 de Novembro de 1999 / November 22 - 26, 1999 Águas de Lindóia, São Paulo,

PARALLEL EIGENSTRUCTURE ASSIGNMENT VIA LINEAR QUADRATIC DESIGN AND MULTIOBJECTIVE GENETIC ALGORITHM WITH DOUBLE FITNESS FUNCTIONS

João V. da Fonseca Neto
Email:jviana@dmcsi.fee.unicamp.br
DEEE/CT-Universidade Federal do Maranhão
65.080-040 - São Luís, MA, Brasil
Celso P. Bottura
Email: cpb@turing.unicamp.br
LCSI/DMCSI/FEEC-Universidade Estadual de Campinas
13.083-970 - C.Postal 6101 - Campinas, SP, Brasil

Abstract. *A strategy for Eigenstructure Assignment based on Linear Quadratic Design and Parallel Multiobjective Genetic Algorithm is presented in this paper. The main contribution of this work is to present the development of the strategy to guide a GA-optimizer based on Pareto optimality, niche induction method, progressive articulation of preferences and the utilization of two different fitness functions. The final design result is a state feedback controller that provides the required Eigenstructure and whose performance is verified using an aircraft state space model.*

keywords: *Optimal control, multiobjective optimization, evolutionary computation, intelligent control, decisionmaking units.*

1. INTRODUCTION

Eigenstructure Assignment (*EA*)plays an important role in control systems engineering, because of its importance on systems response(D'Azzo and Houpis, 1995). *EA* methods have been developed and employed to set up aircrafts flight controls (Clark and Davies, 1997), to adjust controller gains of chemical plants (Liu and Patton,1996), etc. However, when the linear quadratic regulator (*LQR*) design is applied the great challenge is to find out Q and R weighting matrices that lead to the specified requirements. A strategy based on multiobjective optimization, genetic algorithm and parallel computation, which together constitute a parallel multiobjective genetic algorithm (*PMOGA*), was developed to overcomethese difficulties.

A state feedback controller is the final design result whose performance is verified on an aircraft state space model.

The strategy works as follows. Initially, the multiobjective genetic algorithm ($MOGA$) based on Pareto optimality, niche induction method and progressive articulation of preferences (Fonseca and Fleming, 1998a) and (Fonseca and Fleming, 1998b), searchs for Q and R matrices that are used to calculate the controllers gains, which are provided by the Algebraic Riccati Equation (ARE) solution. After, the loop is closed and the feedback system eigenstructure is calculated. Finally, the fitness functions are used to evaluate whether the current EA matches the required eigenstructure.

The whole search process continues on a parallel way in the sense that several EA trials happens in several processors of a workstations network and the best solutions are exchanged during the search to improve the solutions (individuals) and to limit the search space in sectors for each processor near the process end. Two fitness functions are involved in the search , one based on the inequalities method (Zakian and Al-Naib, 1973) and the other directly based on the required EA.

2. PROBLEM SOLUTION FRAMEWORK

The Eigenstructure Assignment problem is formulated as an optimization problem. The entire optimization process has the main purpose of finding state feedback controllers that match the eigenvalues range and eigenvectors specifications.

The optimization method developed is based on the linear quadratic regulator (LQR) design, that gives state feedback controllers, and on a genetic algorithm (GA) optimizer, that searches for the required weighting Q and R matrices.

This section is divided in two parts. The first one shows the LQR design method formulated directly in terms of an multiobjective optimization problem. The second one presents the GA-optimizer features that were developed to perform an intelligent search on the solution space. Details of the problem formulation and GA-optimizer can be seen in (Bottura and Fonseca Neto, 1999a)

2.1. EA-LQR Formulation

Considering a state variable model of a controllable linear invariant system:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{1}$$

where x is the state vector $(n \times 1)$, u is the control vector $(m \times 1)$ and y is the output vector $(r \times 1)$, the control vector is given by :

$$u = -K(Q, R)x \tag{2}$$

where K is the gain matrix $(m \times n)$, Q is the positive semi-definite state weighting matrix, R is the positive semi-definite control weighting matrix.

The controller gain matrix K is obtained by the solution of the algebraic Riccati equation for the linear quadratic optimization problem. The control law is found by the minimization of the quadratic performance cost subject to a restriction $\dot{x} = Ax + Bu$.

The closed-loop system is obtained by substitution of the linear quadratic control law (2) in (1):

$$\dot{x} = (A - BK(Q,R))x \tag{3}$$

The *EA* problem is solved if the eigenstructure of closed-loop system (1) matches the design specifications. Then, the eigenstructure assignment problem can be posed as the determination of the $K(Q,R)$ gain matrix that satisfies the system's performance requirements.

From the above discussions, the optimization problem formulation is stated by gathering the linear quadratic solution and the eigenstructure restrictions:

$$\min_{Q,R} \sum_{i=1}^{n} s_i(Q,R) \tag{4}$$

s.t.

$$s_i(Q,R) \leq 1 \quad i = 1, ..., n \tag{5}$$

$$\lambda_{ei} \leq \lambda_{ci}(Q,R) \leq \lambda_{di} \quad i = 1, ..., n \tag{6}$$

where λ_{ei} and λ_{di} are the left and the right i-ths eigenvalues bounds, respectively, for the i-th desired eigenvalue λ_{ci}. s_i is the i-th normalized eigenvalue sensitivity $s_i(Q,R) = (\frac{\|L_i(Q,R)\|_2 \|R_i(Q,R)\|_2}{< L_i(Q,R) R_i(Q,R) >})/\epsilon_i$; ϵ_i and the i-th design specification. $\|L_i(Q,R)\|_2$ and $\|R_i(Q,R)\|_2$ are the 2-norm of the left and right eigenvectors, respectively, and $< L_i(Q,R) R_i(Q,R) >$ is the eigenvectors dot product.

2.2. *GA*-optimizer

The *GA*-optimizer is a biased stochastic search engine. Its main purpose is to find Q and R matrices that satisfies the design requirements. The *GA*-optimizer starting points are the matrices Q and R set (*population*). This set is randomly determined and it is evolved by the optimizer during the search cycle.

GA-optimizers were specially developed by (Bottura and Fonseca Neto, 1999*a*) and (Bottura and Fonseca Neto, 1999*b*) to handle the *EA* problem when it is formulated as proposed here based on the LQR design and on the inequalities method (Zakian and Al-Naib, 1973). In this subsection a brief description of this optimizer main features is given.

This *GA*-optimizer basic elements are the genetic operations and its fitness function structure. The genetic operations are the manners that the optimizer evolves the matrices set of initial solutions into better directions. The fitness function structure encompass *ARE* solutions, the controllers matrices gains, closed-loop systems eigenstructure and evaluation of the eigenstructure goodness.

The Q and R matrices are modeled as single chromosomes. The elements of these matrices are represented as alleles. Operations between alleles Q and R are not allowed in this implementation.

The genetic operations implemented were crossover, mutation and reproduction. The crossover (*x-over*) operation combines alleles from different solutions (QR-individuals), the

result is two offsprings (two solutions) and this combination degree varies according to the population age. The mutation operation is a modification in the QR-individuals' alleles. The reproduction operation is the duplication of some good QR-individuals. Besides those three operations a new operation was implemented to untrack populations from saturation's levels or two allow a better exploration of the search space; this operation is called guest and it is the insertion of randomly generated individuals into the population.

3. THE PARALLEL *MOGA* ALGORITHM

The GA-optimizer and a decision making unit (DMU) are the basic elements in this multiobjective genetic algorithm ($MOGA$) approach. The optimizer is in charge of QR-individuals search and the DMU is a logical unit that is designed to guide the optimizer into better searches. References (Bottura and Fonseca Neto, 1999*b*) and (Bottura and Fonseca Neto, 1999*c*) give details of the $MOGA$ and DMU algorithm features. This section presents the parallel $MOGA$ definitions and its algorithm basic steps.

The $MOGA$ engine is formalized as a GA-optimizer and DMU pair to perform sequential explorations of the search space. The $MOGA$ is defined as:

$$MOGA = (GA - optimizer, DMU) \tag{7}$$

A parallel $MOGA$, called $PMOGA$, is a search engine that comprises a $MOGA$ set to perform parallel solutions explorations in a distributed environment. The parallel $MOGA$ is defined as:

$$PMOGA = (MOGA_0, MOGA_1, MOGA_2, \ldots, MOGA_n) \tag{8}$$

where $MOGA_0$ is the $MOGA$ coordinator and the $MOGA_i$, $i = 1, \ldots, n$ are $MOGA$s coordinateds.

The algorithm basic steps can be described as follows:

1. The $MOGA_0$ generates randomly an initial population of QR-individual and makes its evaluation according to some fitness function structure.
2. The $MOGA_0$ distributes this initial population among the $MOGA_i$, $i = 1, \ldots, n$.
3. $MOGA_i$, $i = 0, \ldots, n$ start the search space exploration with his $GA - optimizer_i$, $i = 0$ guided by its DMU_i, $i = 0$.
4. When the search cycle reaches its stopping criteria, The permanent population (*solution set*) from $MOGA_i$, $i = 1, \ldots, n$ is sent to the $MOGA_0$.
5. After receiving the $MOGA_i$, $i = 1, \ldots, n$, *solution set*, the $MOGA_0$ selects the best QR-individuals and performs a new operation to find a better *solution set* and to increase the final population profile.

4. The *DMU* strategy

This section presents strategies based on niche induction methods, Pareto optimality and progressive articulation of preferences. The decision making unit is the logical

locus that incorporates the intelligent part of the *MOGA* algorithm. In a recent work (Bottura and Fonseca Neto, 1999*c*) developed a knowledge based procedure for a *DMU* that is randomly triggered by rules. This *DMU* is based on the schema theory and multi-armed bandit paradigm, (Koza, 1992), (Goldberg, 1989) and (Holland, 1975), and designer knowledge of the problem.

The concept of Pareto optimality is used to guarantee that the next feasible solution will be as good as the last feasible solution, i.e., a new solution is considered as the best solution if all new restrictions are smaller than the correspondent old ones.

The concept of progressive articulation of preference is used to guide the search in finding the QR-individuals that first satisfy the hardest restriction and when it is found the *DMU* authorizes the *GA*-optimizer to find a solution that satisfies the second hardest restriction and the process goes on that way.

The niching induction method developed in this work is present in subsection 4.1.. In this proposed method, the individuals whose correspondents alleles are in a well defined range and produce a well defined range of fitness function values are candidates to be part of a natural niche and a mating restriction criterion is used to guide the induced niche exploration.

4.1. Niching induction method

The search space S is divided in S_k search spaces . One of them is explored by the initial population and the others $k-1$ are explored by initial population individual plus an uniform increment. The frontier line are all QR-individual where 100% of their alleles belong to the incremented initial population.

Considering the search space S as the set of all QR-individuals whose alleles are in a specified range, the entire search space is defined as:

$$S = \{QR - individual \quad | \quad 0.001 \leq QR_{allele_i} \leq \infty, \quad i = 1, \ldots, m^2 n^2\} \tag{9}$$

where m is the number of Q alleles and n is the number of R alleles.

The entire search space can be represented as the union of all S_k spaces:

$$S = \{S_1 \cup S_2 \cup S_3 \cup \ldots S_k\} \tag{10}$$

where each S_k space, based on 9, is formalized as:

$$\begin{aligned} S_l \quad = \quad \{QR - individual + \Delta_l \quad | \quad 0.001 \leq QR_{allele_i} \leq \; X_{initial} + \Delta_k l, \\ i = 1, \ldots, m^2 n^2, \quad l = 1, \ldots k\} \end{aligned} \tag{11}$$

Once established the S_k frontiers, the *GA*-optimizer starts the search cycle. Before, the optimizer starts each search, the *DMU* is monitoring for individuals that possesses similar characteristics and that have their fitness function values inside of a defined range. When the *DMU* detects this feature, a niche is found and the next step is this niche exploration.

The niche exploration is done by the *GA-optimizer* and the *DMU* guides this exploration using mating restriction that guarantees the niche population diversification, in

the sense that *x-over* operations are not allowed between niche's individual that are too close.

5. RESULTS

The performace of the *PMOGA* algorithm implementation was studied on a distributed computational environment. The *PMOGA* implementation searchs for feedback controllers that makes the desired eigenstructure assignment. A state space model of a Lockheed aircraft, L1011 Tristar type, linearized to cruise condition, is used as a test system for the proposed *DMU* strategies and its A, B and C matrices, sensitivities restrictions and eigenvalues ranges can be found in (Davis and Clarke, 1995) and (Sobel and Shapiro, 1985).

The performance analysis is made considering a search performed by the *GA*-optimizer without *DMU* strategies actions and when these actions are ready to be activated. The behaviours of the populations for the best individuals, during each step of the search cycle, are presented in Figures (1) and (2). The Figure (1a) shows that the *DMU* strategies really helped the optimizer in finding the feasible solution, while the optimizer without *DMU* provided some improvements in the beginning of the search cycle, but when it reached near population 70 the optimizer tracked and remained at the tracked point until the task ended. It can be seen, Figure (1a), that the optimizer without these *DMU* strategies could not satisfy the *EA* requirements. The E^k plot, Figure (1b), shows that the Pareto's set did occur only until population 50, because searches for Pareto's sets is not the *DMU* highest priority.

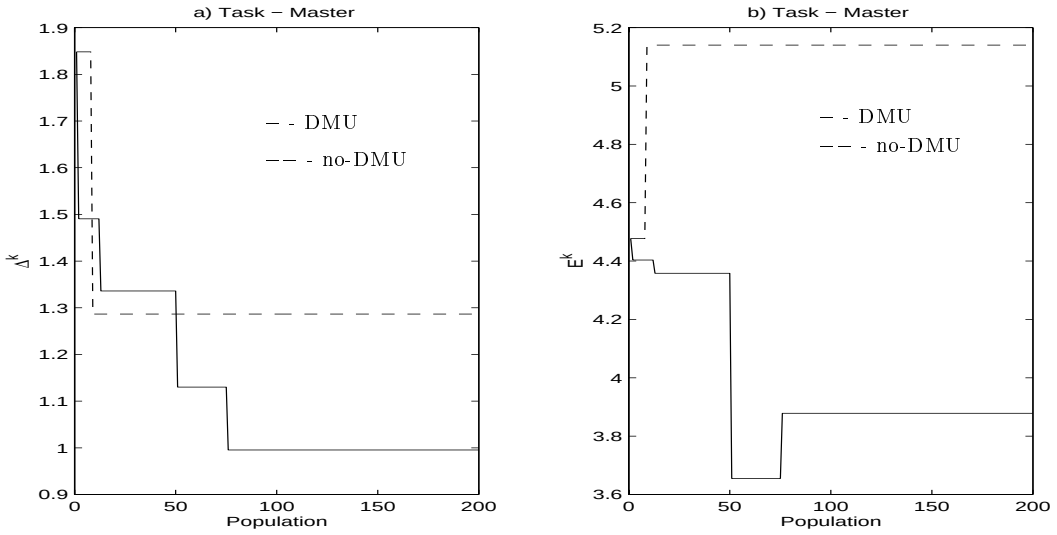

FIGURE 1: Populations × behaviour of the best individual - a) Individual maximum sensitivity S_i and b) Cost funcion E^K

The progressive articulation of preferences is applied to satisfy constraints related with eigenvalues sensitivities s_2^k and s_6^k. As the sensitivity s_6^k for most of the cases is harder to reach its feasibility than sensitivity s_2^k, based on the designer knowledge s_6^k has higher priority than s_2^k. Figure (2) shows that those two are the hardest ones, while the others (s_1^k, s_3^k, s_4^k and s_5^k) have well defined behaviours and the application of articulation

of preferences is not of great importance for them.

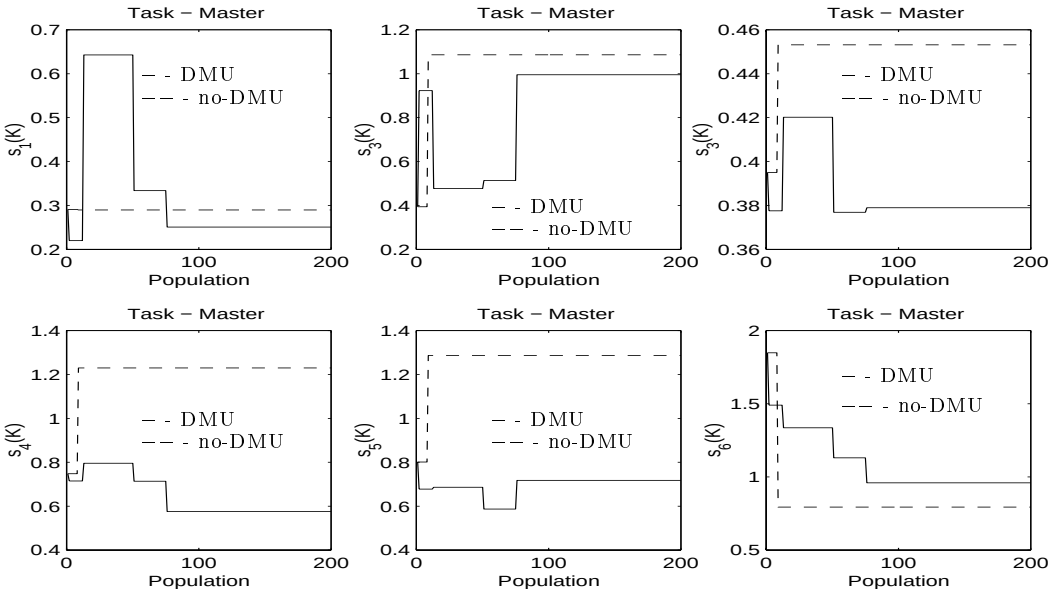

FIGURE 2: Populations $\times$ behaviour of the sensitivities s_i^k

Comparing the final populations profiles, Figure (3), it can be seen that the one evolved by *DMU* strategies has presented a better profile, in an overall manner, than the ones evolved by the canonical *GA*-optimizer, but both optimizers have produced good improvements when compared with the initial population. The eigenvalues maximum sensitivity s_i^{max}, Figure (3a), shows that most of the individuals that come from the *DMU* strategies present a better profile than the ones that come from no-*DMU* strategies. The cost function E^k average is lower for the population evolved by the optimizer with implemented*DMU* strategies.

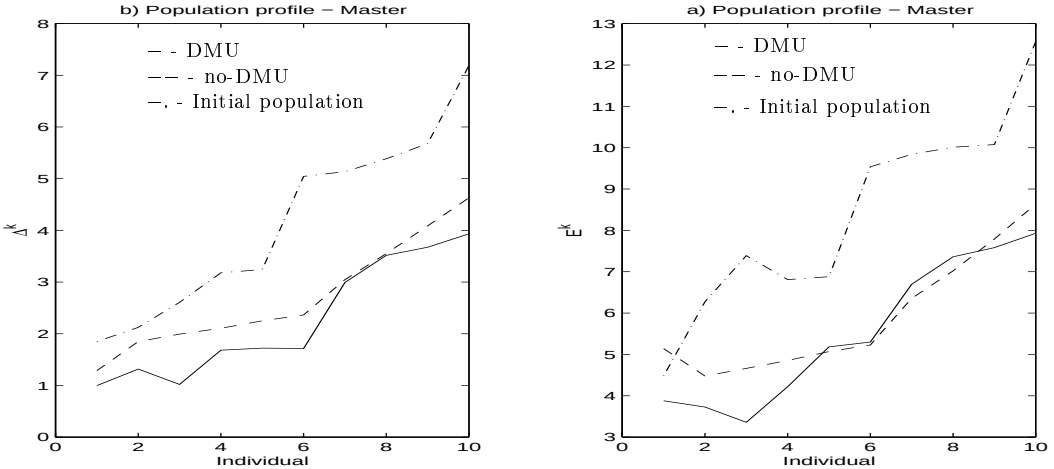

FIGURE 3: Populations profiles $\times$ individual's behaviour - a) Individual maximum sensitivity S_i , b) Cost funcion E^K

The state feedback controller efficiency, Table (1), is observed when it is implemented into the aircraft linearized model and its performance is studied for the impulse signal. Figure (4a) presents the system impulse response for the basic controller that was obtained in reference (Davis and Clarke, 1995); the system impulse response for the controller obtained by the *GA*-optimizer with the *DMU* is presented in Figure (4b). As can be seen the controller designed with the proposed technic presented a very good performance and has the ability to perform the required eigenstructure assignment. The system eigenvalues are inside the required range and the respective eigenvectors satisfy the sensitivity limits.

TABLE 1: Master Controller gains

Task	Gains
Master DMU	0.238 0.003 -0.017 -1.042 -0.023 1.355
	0.007 0.183 -0.914 -1.047 -1.864 3.620

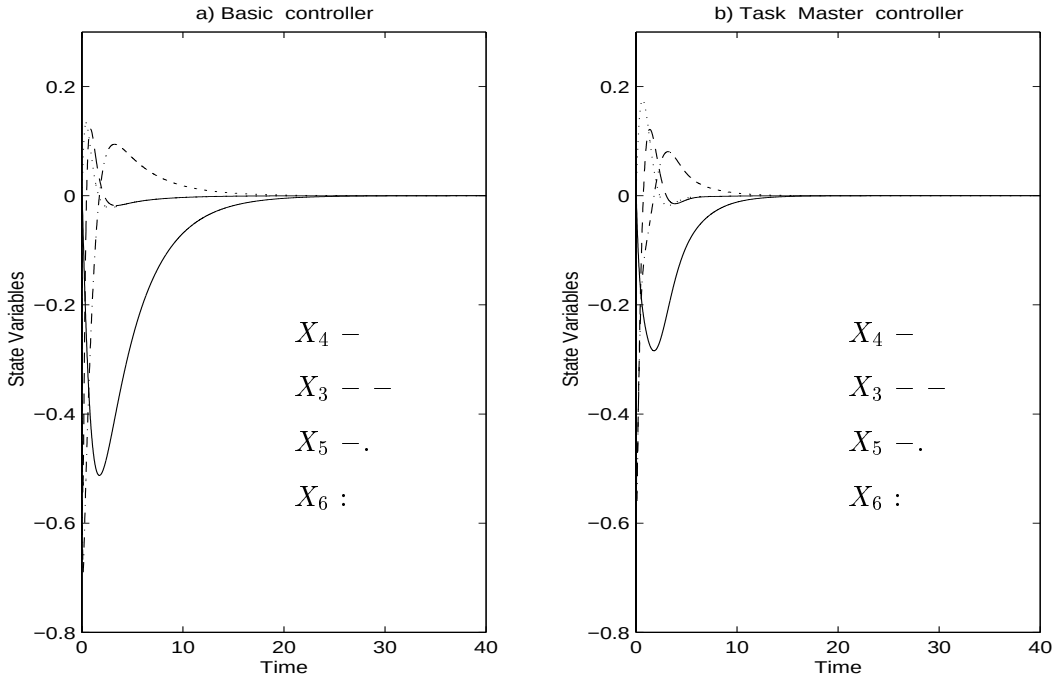

FIGURE 4: System impulse response - a) Basic controller b) Master controller with *DMU* actions

In spite of the slave controller presenting a better performance than the best controller obtained from the *GA*-optimizer without *DMU* strategies, it did not satisfy the *EA* requirements. The design last step is a sequential processing where the individuals of the initial population are the best controllers obtained from the master and slave tasks; this task tryes to minimize the cost function $\sum_{i=1}^{n} f_{\lambda i}(\lambda_{ei} - \lambda_{ci})^*(\lambda_{ei} - \lambda_{ci}) + (\vec{v}_{ei} - \vec{v}_{ci})^* F_{vi}(\vec{v}_{ei} - \vec{v}_{ci})$, where λ_{ei} i-th specified eigenvalue, λ_{ci} i-th desired eigenvalue, f_i i-th eigenvalue weighting, $\vec{v}_{ei}$ i-th specified eigenvector, $\vec{v}_{ci}$ i-th desired eigenvector, F_{vi} i-th eigenvector weighting diagonal matrix. Aiming to get better controllers and/or to improve the controllers population profile. The sequential task could not get a better controller than the one obtained

by the parallel LQR designs, but improved the population profile. Two reasons can justify these events occurrence; first: the initial population was not a good starting point and second: the eigenvalues limits were so tight that it is very hard to find a controller that satisfies both the eigenvectors restrictions and the eigenvalues specified ranges.

6. CONCLUDING REMARKS

The parallel multiobjective genetic algorithm ($PMOGA$) comprised by a set of GA-optimizer and DMU pairs working together on a ordinary distributed computational environment has shown to be a valuable controller design tool to perform searches for matrices Q and R of the LQR design that satisfies a required eigenstructure.

An advantage of the proposed technic is the small amount of time spent to find the state feedback controller that satisfies the required EA; approximately 1.53 minutes for the worst case on a simple distributed environment with two tasks (a master and a slave) searching for two controller families. Another advantage is that the proposed technic incorporates the qualities of linear quadratic design (LQR). A disadvantage is the choice of a bad initial population (starting solution points) that can make the search very hard or even unable to find a feasible solution. The next paragraph proposes alternative solutions for this disadvantageous situation.

The first proposal is: DMU actions improvement by increasing the number of individuals in the permanent population; if this population is increased the number of niches can be enlarged, as well as the number of individuals that belong to a given niche, allowing a better $PMOGA$ exploration of the search space, where the starting point is the same controllers family. The second proposal is to find a better starting point with the following features: the controllers eigenvalues are on the specified ranges or the hardest sensitivity restrictions are very close to their satisfaction.

REFERENCES

Bottura, C. P. and J.V. da Fonseca Neto (1999*a*). Parallel eigenstructure assignment via LQR design and genetic algorithms. *1999 American Control Conference - ACC99 - San Diego - CA - USA.* pp. 2295-2296.

Bottura, Celso P. and J. V. Fonseca Neto (1999*b*). Parallel genetic algorithm fitness function team for eigenstructure assignment via LQR designs. *Congress on Evolutionary Computation - CEC99. Washington, DC, USA. Vol* **2**, 1035-1042.

Bottura, Celso P. and J. V. da Fonseca Neto (1999*c*). Rule-based decision making unity for eigenstructure assignment via parallel genetic algorithm and LQR designs. *Submitted.*

Clark, T. and R. Davies (1997). Robust eigenstructure assignment using the genetic algorithm and constrained state feedback. *IMechE* **211**(Part–I), 53-61.

Davis, R. and T. Clarke (1995). Parallel implementation of a genetic alghorithm. *Control Eng. Practice* **3**(1), 11-19.

D'Azzo, J. . and C. H. Houpis (1995). *Linear Control Systems, Analysis and Design - Conventional and Modern.* fourth ed.. Mc Graw Hill Inc. USA.

Fonseca, C.M. and P.J. Fleming (1998*a*). 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* **28**(1), 26-37.

Fonseca, C.M. and P.J. Fleming (1998*b*). Multiobjective optimization and multiple constraint handling with evolutionary algorithms-part II: Application example. *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans* **28**(1), 38-48.

Goldberg, David Edward (1989). *Genetic Algorithms in search, optimization, and machine learning*. Addison-Wesley Publishing Company Inc.. USA.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press . Ann Arbor-Michigan-USA.

Koza, John R. (1992). *Genetic Programmming : On the Programming of Computers by Means of Natural Selection*. The MIT Press. Cambridge, Massachusetts - USA.

Liu, G.P. and R.J. Patton (1996). Robust control design using eigenstructure assignment and multiobjective optimization. *International Journal of Systems Science* **27**(9), 871-879.

Sobel, K. M. and E. Y. Shapiro (1985). Eigenstructure assignment : A tutorial part i theory. In: *Proceedings of American Control Conference* 5^{th}. Vol. 1. Saint-Nazaire, USA. pp. 456-460.

Zakian, V. and U Al-Naib (1973). Design of dynamical and control systems by the method of inequalities. *IEE-Proceedings* **120**(11), 1421-1427.