Proceedings of the American Control Conference
Chicago, Illinois • June 2000

Rule-based Decision-making Unit for Eigenstructure Assignment via Parallel Genetic Algorithm and LQR Designs

Celso P. Bottura † and João V. da Fonseca Neto‡
†LCSI/DMCSI/FEEC/UNICAMP -C.Postal 6101 13.083-970 - Campinas - SP - Brazil
bottura@dmcsi.fee.unicamp.br
‡DEE/CT/ Universidade Federal do Maranhão - C. Bacanga 65.080-040 - São Luís - MA - Brazil
jviana@dmcsi.fee.unicamp.br

Abstract

This paper proposes a decision-making unit rule-based genetic algorithm for eigenstructure assignment via *LQR* designs. The proposed decision making framework is formulated in terms of schema theorem, multi-armed bandit problem and rule-based knowledge; the purpose of these three elements is to guide, in an expert manner, the parallel genetic search to find feedback controllers families that perform the specified eigenstructure assignment. The design techniques are based on multiobjective optimization and independent runs parallel genetic algorithm. An aircraft state space model is used to study the strategy performance.

1 Introduction

One of the main features of intelligent systems may be human knowledge emulation by a decision making unit. In a previous work, [1], we observed that a regular GA could lead the search into unfeasible regions of the solution's space. To improve the GA's searching power in [2], we proposed a decision making unit (DMU) with strategies to guide the search to increase the likelihood of finding feasible solutions. During the search process it was observed that the insertion of the designer's knowledge and decisions guided the genetic search to obtain good controllers families. Motivated by improvements obtained with the designer's inference, a rule-based decision-making unit to guide the genetic search is here proposed. The results obtained from the search are used to calculate state feedback controllers gains for eingenstructure assignment via *LQR* designs. The procedure developed in this work is part of a second step in the direction of an intelligent system for multivariable control design methodology we are proposing for eigenstructure assignment.

The Eigenstructure Assignment problem has been focused by researchers because of its great influence on dynamic systems response. Deterministic methodologies and techniques have been developed, [5], [10], and many others, to solve the *EA* problem, but only in the nineties some effort has been spent to make this assignment via genetic algorithms, [4] and [9]. The feasibility of this kind of solution ocurred as a consequence of faster and cheaper *CPU's* giving chance for new solution's tools, [8], [6] and [7].

The great difficulty to find Q and R weighting matrices for *LQR* design, that can satisfy *EA* requirements, leads to a framework development based on multiobjective optimization and parallel genetic algorithm to solve this problem.

This work is organized as follows. Section (2) describes the *EA* problem formulated as a multiobjective optimization problem. Section (3) presents the parallel multiobjective genetic algorithm (*PMOGA*) definitions and how it works on a distributed environment. Section (4) presents the main features of the genetic algorithm optimizer. Section (5) presents the rule-based *DMU* framework, describing its basic elements and strategies. Section (6) presents *DMU* and controllers performances results obtained from simulations on a high performance computer network and section (7) presents the conclusions.

2 Problem formulation

The eigenstructure assignment via classic linear quadratic optimization problem is formulated as a multiobjective optimization problem and its details can be found in [1].

The controller gains $K(Q,R)$, where Q and R matrices are independent variables, are given by the Algebraic Riccati Equation (*ARE*) solution's for the *LQR* problem, where the control law $u = -K(Q,R)x$ is found when the minimization of the quadratic performance cost $J = \int_0^\infty [x^T Q x + u^T R u] dt$, subject to the restric-

0-7803-5519-9/00 $10.00 © 2000 AACC

tion $\dot{x} = Ax + Bu$ is performed.

Clearly, the eigenstructure assignment problem, from this point of view, consists on the gain matrix $K(Q,R)$ determination that imposes the specified closed-loop system $\dot{x} = (A-BK(Q,R))x$, where its spectrum range must satisfy the design specifications and the left and right eigenvectors must satisfy eigenvalue sensitivity restrictions.

The multiobjective optimization problem (MOP) formulation that allows the determination of a controller $K(Q,R)$ through application of biased random search technics to solve the eigenstructure problem, is obtained by joining the LQR problem solution and the eigenstructure restrictions (closed-loop system and eigenvalue sensibility spectrum bounds). The MOP formulation in a normalized form is:

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

s.t.

$$s_i(Q,R) \leq 1 \quad i = 1,...,n \qquad (2)$$

$$\lambda_{li} \leq \lambda_{di}(Q,R) \leq \lambda_{ri} \quad i = 1,...,n \qquad (3)$$

where $s_i(Q,R) = (\frac{\|L_i(Q,R)\|_2 \|R_i(Q,R)\|_2}{<L_i(Q,R),R_i(Q,R)>})/\epsilon_i$ is the normalized i-th eigenvalue sensitivity and the i-th design specification $\epsilon_i > 0$; $\|L_i(Q,R)\|_2$ 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. λ_{li} and λ_{ri} are the left and the right i-ths eigenvalues bounds, respectively, for the i-th desired eigenvalue λ_{di}.

3 Parallel Multiobjective Genetic Algorithm

The multiobjective genetic algorithm ($MOGA$), Fig 1, was designed based on the interaction between the decision-making unit and the genetic algorithm (GA) optimizer. A $MOGA$ set interacting on a distributed enviromment constitutes the parallel multiobjective genetic algorithm, $PMOGA = (MOGA_c, MOGA_1, MOGA_2, ..., MOGA_n)$, where $MOGA_c$ is the coordinator or master task and the $MOGA_i$, $i = 1,\ldots,n$, are the coordinateds or slave tasks. The $PMOGA$ basic elements description are presented in [2].

In a few words, the $PMOGA$ works as follows: the DMU's are fed by an initial population (randomly generated); this population after an evaluation by the DMU coordinator (DMU_C) is sent to the distributed GA-optimizers, $OTIM_D = (GA_1, GA_2, ..., GA_n)$, where each GA_i makes one search and generates a transient population to feed their own DMU_i, the set of distributed DMU's is $DMU_D = (DMU_1, DMU_2, ..., DMU_n)$. Each DMU_i takes decisions to guide the search, assembles a new population and sends it to its GA_i optimizer. These processes go on until a stopping criterion is reached.

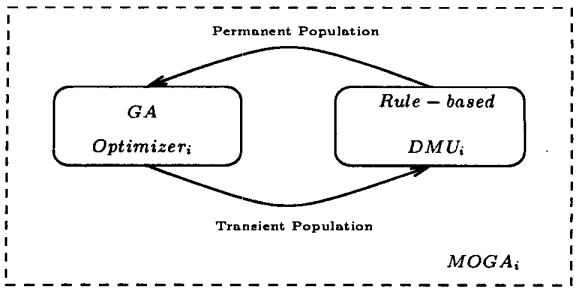

Figure 1: $MOGA_i$ basic strucuture as $PMOGA$ single unit.

4 Genetic Optimizer

This section gives a general description of the GA-optimizer basic elements, that are: matrices Q and R genetic modeling, genetic operations and fitness function team (calculations and ordering). Further details can been seen in [2]. The matrices modeling and genetic operations are all numerically performed on a decimal basis.

The GA-optimizer main features are the guest operator, whose purpose is to untrack some populations of saturation levels and to allow a better exploration of the search space, and small size permanent and transient populations.

Each matrix Q and R is represented by one chromosome and matrices Q and R pairs are called QR-individuals and a set of QR pairs comprises a population.

The genetic operations are performed in two sequential steps, that are: chromosomic operations and evaluations of individuals generated from those operations. The first step, chromosomic operations are reproduction, crossover, mutation and guest. The second step, fitness functions calculation and ordering provide information to the decision-making unit.

The fitness function set: $FF_{team} = (ff_1, ff_2, ..., ff_n)$ defines a fitness function team. Each ff_j, $j = 1,...,n$ structure is built up with one cost function and a previous selection criterion.

5 Rule-based Decision-Making Unit

The decision-making unit (*DMU*) proposed in this work is a logical rule-based framework, whose main purpose is to formulate decision strategies to guide the genetic optimizer search. The decisions are made based on genetic operations past history, each one obtained from *GA*-optimizer fitness function structure team, schema theorem, multi-armed bandit paradigm and designer knowledge, that are inserted interactively to break *DMU* static rules and to furnish new directions to the search.

A strategy based on the schema theorem, to explore the individual's maximum strength, and on the multi-armed bandit paradigm, to extract strength potentialities that can exist on the weakest one, is presented in [3] and these strategies are translated into *boolean* rules and applied to this *DMU*.

The inserted rules, R_{age} and Q_{age} rules, act directly on the crossover (*x-over*) operations and their control parameters are the *x-over* multiplicative factors that were designed primarily to suffer variations according to population's age and that now have as a second function: to guide the *GA*-optimizer search. Initially, the individual matrix R was chosen to act as a regulation vehicle because of its influence in feedback controller determination due to its direct participation on the algebraic Riccati equation and on controllers gains calculations. After exhaustive computational experiments we concluded that dynamic changes on the parameter q, that acts on matrix Q alleles during *x-over* operations, contributed to improve the *GA*-optimizer search, not only reducing the Q and R matrices search cycle but also improving the controller quality.

The R_{age} and Q_{age} rules purposes are to define the control parameters r_{age} and q_{age} directions (left or right) and sizes variations. The r_{age} and q_{age} variations are based on a comparative analysis between individuals R, controllers gains K, eigenvalues sensitivities and eigenvalues ranges restrictions.

The rule-based *DMU* basic structure and its interactions with the *GA*-optimizer are shown on Fig. 2, that shows *DMU* interacting only with *x-over* operations for R_{age}-rule. The schema theorem (*STheorem*) and the multiarmed bandit paradigm (*MABParadigm*) act on both R and Q individuals. The rules will act based on probability and the R_{age}-rule will actively participate on the search if any *MOGA* is having problems to find feedback controllers. The time spent with the analysis justifies this strategy. The *STheorem* and *MAB-Paradigm* strategies actions are based on three rules. The first one acts on population or best individual premature convergence detection. The second one acts based on small probability of occurrence. The third

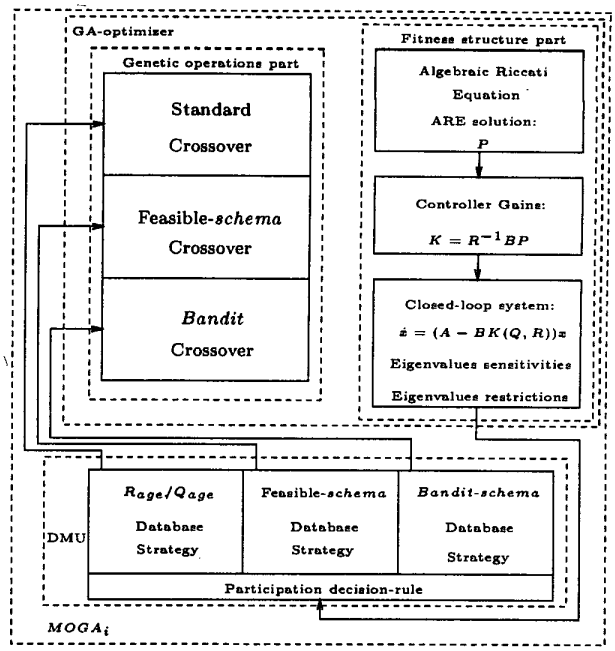

Figure 2: Interactions between Rule-based *DMU* and *GA*-optimizer in a $MOGA_i$

one acts in a deterministic manner and periodically between generations life and death. The difference between these rules is their occurrence probability.

6 Results

The results were obtained from simulations performed on a computer network. The simulated system was an aircraft's, L1011 Tristar type, state space variable linear model; its A, B and C matrices are given in [10], and the design specifications are given in [4].

Ten search tasks were spawn on a high performance parallel computation environment, that is based on IBM-SP computers. One of the tasks (coordinator) creates the initial population and distributes it among the other tasks (coordinateds). The permanent population is built up of 10 individuals (controllers), i.e., for each new generation only 10 individuals survive to the conditions imposed by the fitness structure functions.

The simulation's main purpose was to verify the R_{age} and Q_{age} rules performance and these intentions were made possible due to the choice of a hard case, as it was difficult to find a feedback controller satisfying eigenvalues ranges restrictions or eigenvalues sensitivity specifications.

Fig. 3 presents the changes on *x-over* parameters pro-

vided by the rules and the worst eigenvalue sensitivity $s_i(Q,R)$ for eight tasks out of ten; the main purpose of these figures is to show the effectiveness of the proposed rules for a hard convergence case. Each figure shows, in a comparative manner, the parameters variations and the eigenvalues sensitivity restrictions for the same tasks, both when the rules are activated and for the case that the rules are not triggered by the *DMU*. For instance, Fig. 3, presents four curves (a, b, c and d); Fig. 3a shows the parameters changes over intervals and there are no modifications of the parameters inside of each interval; Fig 3b presents the worst eigenvalue sensitivity for these parameters variations; Fig. 3c shows the parameters variations according to the R_{age} and Q_{age} rules and there are parameters changes on each interval; Fig. 3d presents the corresponding $s_i(Q,R)$, but the worst one, Δ_k, obtained for these rules actions. The final population profile can be trans-

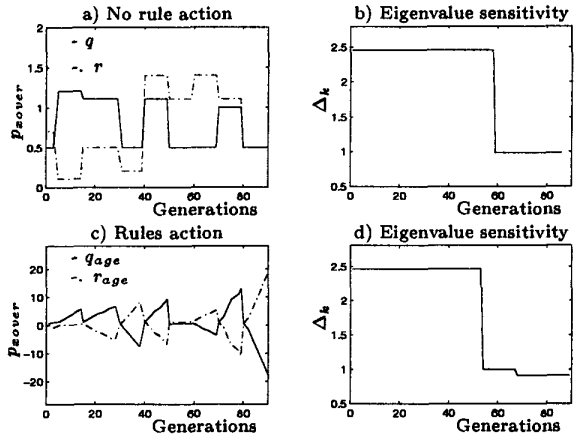

Figure 3: Master task - *x-over* parameters changes and worst eigenvalue sensitivity

lated into a performance qualitative index that represents the quality of the *GA*-optimizer search. Fig. 4 presents the populations profiles by means of the worst eigenvalue sensitivity, Δ_k, eq. (2), and by means of the eigenvalue sensitivity cost function sum, E_k, eq. (1). Comparisons are made with the initial population and the final populations when the *x-over* parameter rules are not on and when these rules are activated. All the obtained population profiles with the implemented *x-over* rules presented a better profile than those without these rules activations.

The controllers performances were tested by impulse signal responses. Two types of simulations were performed. The first one, considers a family of controllers that comes from the same task and all are feasible. The second one takes into account controllers that present the smallest worst eigenvalues sensitivities among all controllers that come from the final population of each

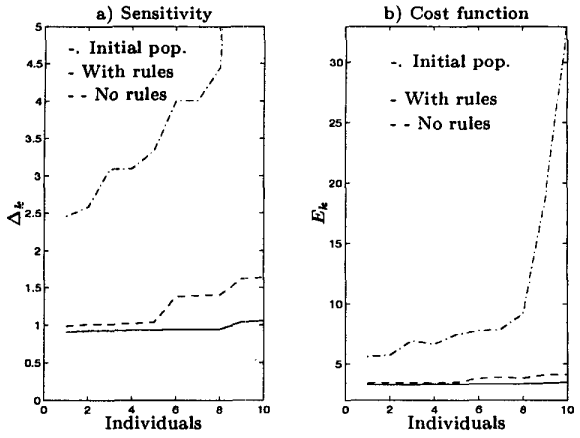

Figure 4: Master task population profile - a) Worst eigenvalue sensitivity. b) Sensitivity cost function.

distributed task, eq. (4). Another simulation, similar to the first one, was performed considering a controllers family where the majority of the feasible controllers presented slight differences among worst eigenvalue sensitivities and also small differences among their eigenvalues ranges.

$$controller_k = \min_j \max_i s_i(Q,R) \qquad (4)$$
$$i = 1,\ldots,n;\ j = 1,\ldots,m;\ k = 1,\ldots,l$$

where n is the dynamic system order, m is the permanent population size and l is the number of tasks. For simulation type one, l is the number of controllers that present the smallest Δ_k.

Fig 5 presents the simulation type one results associated to a controllers family that comes from slave-03 task. These results are compared with the basic controller impulse response simulation, given by [4]. Most of these controllers present good performance, with the exception of controller 3, Fig. 5d, that did not present a desirable performance for the state variable X_4; if more trials had been made a better controller 3 could also have been obtained.

For the type 2 simulation, the controllers performances are shown on Fig. 6. The controllers for most of the tasks were better than the basic controller, except for slave 2 task controller, Fig. 6c.

7 Conclusions

The proposed rule-based decision-making unit, using the schema theorem, the multiarmed bandit paradigm

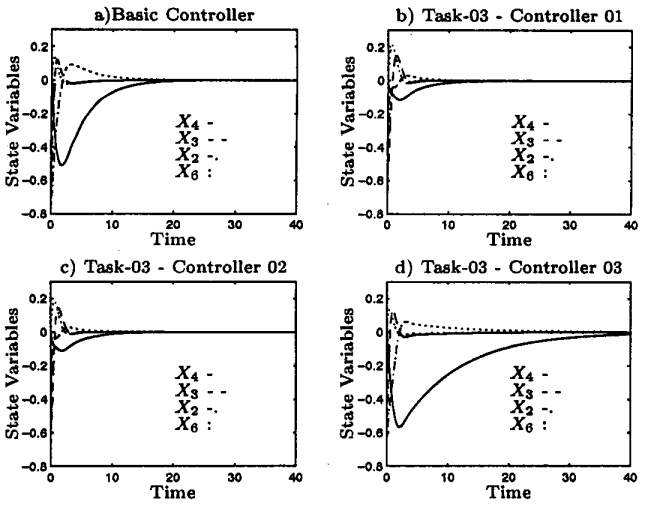

Figure 5: Impulse input signal response - Task 03 controllers 01-02

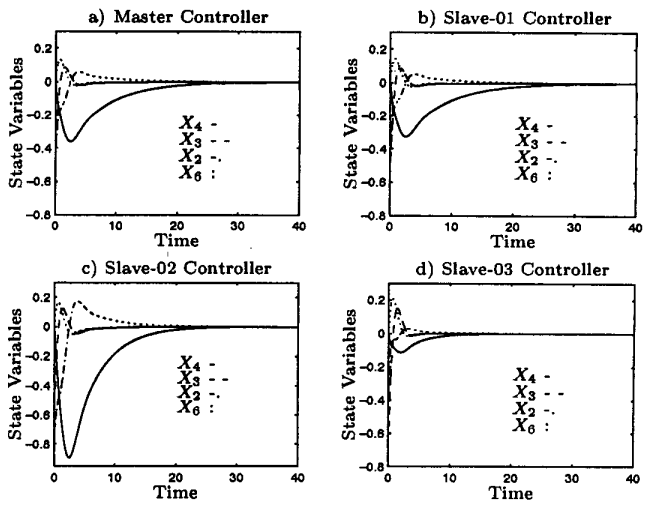

Figure 6: Impulse input signal response - Master task and slaves 01-03 task controllers

and the crossover operation parameters tunning (Q_{age} and R_{age} rules), interacting with the *GA*-optimizer, has shown to be a valuable logical device to guide the genetic search in finding the Q and R weighting matrices for the eigenstructure assignment problem via *LQR* designs.

The DMU rule-based strategies improved the computational efficiency of our parallel genetic algorithm. Its performance can be considered good, taking into account that a hard case was chosen to match the design specifications. In spite of not guaranteing to find a global extrema, a problem that increases when the solution's space frontiers are enlarged, as a problem solver tool for eigenstructure assignment, it can be said that the parallel genetic algorithm together with a fitness functions team, rule-based DMU, small population and guest operator has shown to be an efficient tool for controller design when the entire system eigenstructure has to be assigned.

References

[1] C. P. Bottura and J.V. Fonseca Neto. Parallel Eigenstructure Assignment via LQR Design and Genetic Algorithms. *American Control Conference - ACC99. San Diego, California, USA.*, pages 2295–2296, June 1999.

[2] Celso P. Bottura and J. V. Fonseca Neto. Parallel Genetic Algorithm Fitness Function Team for Eigenstructure Assignment via LQR Designs. *Congress on Evolutionary Computation - CEC99. Washington, DC, USA. Vol*, 2:1035–1042, July 1999.

[3] C. P. Bottura and J.V. Fonseca Neto. The Schema Theorem and Multiarmed Bandit Paradigm Influences on Eigenstructure Assignment via LQR Designs. 4º *Simpósio Brasileiro de Automa cão Inteligente - SBAI99. São Paulo - SP - Brasil*, pages 402–406, September 1999.

[4] R. Davis and T. Clarke. Parallel implementation of a genetic algorithm. *Control Eng. Practice*, 3(1):11–19, 1995.

[5] M. M. Fahmy and J. O'Reilly. On Eigenstructure Assignment in Linear Multivariable Systems. *IEEE Transaction on Automatic Control*, 27(3):690–693, June 1982.

[6] David E. Goldberg. *Genetic Algorithms in Search, Optimization ,and Machine Learning.* Addison Wesley Publishing Company, Ann Arbor-Michigan-USA, 1989.

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

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

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

[10] K. M. Sobel and E. Y. Shapiro. Eigenstructure assignment : A tutorial part i theory and part ii applications. In *Proceedings of American Control Conference* 5^{th}, volume 1, pages 456–460, Saint-Nazaire, USA, 1985.