UNICAMP

Universidade Estadual de Campinas

LCSI Faculdade de Engenharia Elétrica e de Computação
Departamento de Máquinas, Componentes e Sistemas Inteligentes
Laboratório de Controle e Sistemas Inteligentes

Alocação Computacional Inteligente de Autoestruturas para Controle Multivariável

Tese apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do título de

Doutor em Engenharia Elétrica

por

João Viana da Fonseca Neto
Mestre em Engenheira Elétrica - UFPB/PB

Prof. Dr. Celso Pascoli Bottura
Orientador - FEEC/UNICAMP/SP

Aprovada em 10 de março de 2000 pela banca examinadora:

Prof. Dr. Celso Pascoli Bottura (Presidente)

Prof. Dr. Eugenius Kaszkurewicz - COPPE/UFRJ/RJ

Prof. Dr. Ely Carneiro de Paiva - CTI/CAMPINAS/SP

Prof. Dr. Maurício Ferreira Magalhães - FEEC/UNICAMP

Prof. Dr. Fernando José Von Zuben - FEEC/UNICAMP

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

F733a

Fonseca Neto, João Viana da
Alocação computacional inteligente de autoestruturas para controle multivariável / João Viana da Fonseca Neto.--Campinas, SP: [s.n.], 2000.

Orientador: Celso Pacoli Bottura.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Autovalores. 2. Teoria do controle. 3. Controle automático - Sensibilidade. 4. Processamento paralelo (computadores). 5. Algoritmos genéticos. 6. Análise modal. 7. Sistema de controle por realimentação. I. Bottura, Celso Pascoli. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

Alocação Computacional Inteligente de Autoestruturas para Controle Multivariável

Apresenta-se nesta tese uma proposta para alocação de autoestruturas em sistemas dinâmicos lineares multivariáveis por realimentação de estado que tem por base o projeto do regulador linear quadrático (RLQ), otimização multiobjetivo, computação evolutiva e programação paralela. O problema da alocação de autoestruturas é formulado em termos de projetos RLQ e de um método de desigualdades no intuito de colocar a formulação proposta como de problema de otimização multiobjetivo. Este problema é solucionado através do desenvolvimento de algoritmo genético paralelo dedicado à busca das matrizes de ponderação do projeto RLQ. Estratégias de busca são formuladas e elementos de inteligência computacional são utilizados para modelá-las e implementá-las em uma unidade de decisão lógica que interage com o otimizador genético.

Abstract

Computationally Intelligent Eigenstructure Placement for Multivariable Control

In this thesis a proposal for eigenstructure placement for multivariable linear dynamic systems by state feedback based on the linear quadratic regulator (LQR), multiobjective optimization, evolutionary computation and parallel programming is presented. The eigenstructure placement problem is formulated based on LQR designs and on an inequality method to conceive a multiobjective optimization problem for the proposed formulation. The solution to this problem is obtained via the development of a parallel genetic algorithm dedicated to the search of LQR design's weighting matrices. Search strategies are formulated and elements of computational intelligence are used to model them and their implementations are made on a logical decision unit that interacts with the GA-optimizer.

Universidade Estadual de Campinas

LCSI Faculdade de Engenharia Elétrica e de Computação
Departamento de Máquinas, Componentes e Sistemas Inteligentes
Laboratório de Controle e Sistemas Inteligentes

Tese : **Alocação Computacional Inteligente de Autoestruturas para Controle Multivariável**

Autor : **João Viana da Fonseca Neto**

Orientador : **Prof. Dr. Celso Pascoli Bottura**

Aprovada em 10 de março de 2000 pela banca examinadora

Prof. Dr. Celso Pascoli Bottura (Presidente)

Prof. Dr. Eugenius Kaszkurewicz - COPPE/UFRJ/RJ.

Prof. Dr. Ely Carneiro de Paiva - CTI - Campinas/SP.

Prof. Dr. Maurício Ferreira Magalhães - FEEC/UNICAMP.

Prof. Dr. Fernando José Von Zuben - FEEC/UNICAMP.

Dedico este trabalho
à Silvia, ao Murilo e aos meus pais.

Agradecimentos

- Ao Prof. Celso Pascoli Bottura pela orientação, aprendizado, receptividade a novas idéias e estímulo ao longo destes anos;
- Aos colegas de laboratório Maurício José Bordon, Paulo James de Oliveira, Annabell Del Real Tamariz, Rogério Bastos Quirino, Angel Fernando Torrico Caceres, Sérgio A. Augusto Filho, Manoel F. Soares Neto e Gilmar Barreto pela amizade, troca de informações e debates;
- Ao amigo e colega Pablo Moscato, por mostrar-me os algoritmos genéticos e suas potencialidades;
- Aos professores que muito contribuíram para minha formação: Alice Maria B. H . Tokarnia, Vinícius A. Armetano, Maurício F. Magalhães, Vera Maura Fernandes, Ivan L. Marques Ricarte, Paulo M. França, Dilvan Moreira e Wagner Caradori do Amaral;
- Aos professores do DMCSI, em especial Gilmar Barreto e Ioshiaki Doi, pela agradável convivência do dia a dia;
- Ao colegas Eduardo Trettel e Márcio Stocco do DMCSI e Fernando Whitaker, Ana Seixas e Ivan Lima do CENAPAD pelo apoio computacional;
- A Alaíde e D.Edna do DMCSI/FEEC pelas inúmeras ajudas;
- À UNICAMP, de uma forma geral;
- À Universidade Federal do Maranhão, por ter investido na minha formação profissional e aos meus colegas do Departamento de Engenharia Elétrica, pelo incentivo;
- Ao programa CAPES-PICDT, pelo apoio financeiro;

Agradeço à Deus, que me deu a oportunidade de conhecer
todas estas pessoas e que me fez seguir este caminho.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O objetivo desta tese é apresentar uma proposta para resolver o problema de alocação de autoestruturas em sistemas dinâmicos utilizando o projeto do regulador linear quadrático, a computação paralela e técnicas de computação evolutiva.

Classificar, rotular ou contextualizar não é tarefa fácil, principalmente quando certos tópicos estão na fronteira do conhecimento. Inteligência computacional ou ciência computacional?. Onde enquadrar esta pesquisa?. Quando observa-se as técnicas utilizadas para solucionar o problema em questão, esta pode estar enquadrada em ambos os escopos. Antes de assumir posturas de contextualização genérica, o próximo parágrafo procura esclarecer de forma breve o que se entende por inteligência computacional e por ciência computacional.

Atualmente, *ad hoc* início 2000, não existe uma definição aceita, de forma geral, para os termos ciência e engenharia computacional. Contudo, a intersecção de uma ciência aplicada, com a matemática e com a ciência da computação define o campo de atuação da ciência e da engenharia computacional que neste trabalho será simplesmente referenciada como ciência computacional. Esta ciência pode ser considerada como uma metodologia comum para uma série de ciências que utilizam os mesmos tipos de ferramentas. Estes comentários foram extraídos do *Computational Science Education Project*, (CSEP, 1996).

Inteligência computacional e *soft computing* são duas maneiras diferentes de denominar o paradigma utilizado para explorar o potencial de construir máquinas inteligentes através do modelamento do comportamento e de mecanismos de organismos biologicamente inteligentes, (Fogel *et al.*, 1999).

Lógica fuzzy, redes neurais artificiais e computação evolutiva são as metodologias que compõem a inteligência computacional, (Fogel *et al.*, 1999); uma quarta metodologia, chamada de computação probabilística, faz parte do que recebe a denominação atual de *soft computing*, (Bonissone *et al.*, 1999).

Do exposto nos dois últimos parágrafos pode-se concluir que o trabalho proposto pode ser enquadrado na classificação de ciência computacional, bem como na classificação de inteligência computacional.

Se enquadra no contexto de ciência computacional por envolver um conjunto de ciências aplicadas bem definido, tais como: engenharias aeronáutica, química, elétrica e outras relacionadas com o problema de alocação de autoestruturas; porque a matemática contribui em dois aspectos: na formulação de teorias de controle e de computação evolutiva, que permitem estabelecer as bases para a formulação e a solução do problema; e porque a ciência da computação fornece as técnicas de programação paralela, redes de estação de trabalho e os computadores de alto desempenho para resolver o problema.

Se enquadra no contexto de inteligência computacional porque o mecanismo de solução proposto tem por base uma metodologia de computação evolutiva, que imita o processo de evolução natural de organismos biológicos para determinar a solução do problema. Entre as várias técnicas de computação evolutiva, escolheu-se a dos algoritmos genéticos como mecanismo de busca estocástica polarizada para solucionar o problema da alocação de autoestruturas. Os algoritmos genéticos realizam uma busca em paralelo no espaço de solução utilizando uma representação genotípica das possíveis soluções.

Devido à importância do controle multivariável no contexto da teoria de controle e por ser ele objetivo importante desta tese, os próximos parágrafos estabelecem o que se entende por controle multivariável e comenta brevemente o seu desenvolvimento.

A teoria de controle multivariável tem por objeto o controle de sistemas com múltiplas entradas e múltiplas saídas.

Patel e Munro, (Patel e Munro, 1982), traçam um perfil do desenvolvimento do controle multivariável, até 1981, tendo como referência as áreas que mais necessitaram deste tipo de abordagem e, como consequência, mais influenciaram no desenvolvimento desta teoria. A teoria de controle multivariável começou a ser formulada em meados dos anos 1950 e sua evolução teve como inspiração o computador digital com o conceito de programa armazenado e a necessidade de solucionar importantes problemas de controle, principalmente os problemas ligados à indústria aeroespacial, que deram ori-

gem à teoria de controle ótimo, que tem por base a representação dos sistema no espaço de estado, e para a qual as primeiras contribuições, talvez as mais importantes, foram dadas por *Bellman* e *Pontryagin*.

O desenvolvimento inicial da teoria de controle linear multivariável no espaço de estado foi conduzido por duas abordagens, e os trabalhos de *R.E Kalman* são precursores e fundamentais para ambas. Na primeira abordagem, *Kalman* propõe um procedimento para o projeto através do controle linear ótimo com índice de desempenho quadrático, (Kalman, 1960*b*) e (Kalman, 1963). A outra abordagem enfatizou o desenvolvimento de teoria para a análise da estrutura interna dos sistemas lineares e mais uma vez teve *Kalman* como precursor ao desenvolver os conceitos estruturais de controlabilidade e observabilidade, (Kalman, 1960*a*).

A abordagem estrutural serviu de base para o desenvolvimento de novos métodos para projeto de sistemas de controle multivariável. *Rosenbrock*, por uma série de contribuições à teoria de controle multivariável, tais como: ser o precursor da teoria do controle modal, (Rosenbrock, 1962), e desenvolver uma teoria que envolve métodos nos espaços de frequência e de estado, (Rosenbrock, 1970) e (Rosenbrock, 1974) deve ser destacado.

Este capítulo introdutório é organizado da seguinte maneira. A seção (1.1) apresenta a importância do problema da alocação de autoestrutura, salientando a sua origem e uma breve descrição do desenvolvimento das teorias para alocação. A seção (1.2) descreve as técnicas de computação evolutiva e sua utilização na solução de problemas de controle. A seção (1.3) explicita os motivos que levaram a desenvolver esta pesquisa e porque optou-se pelo método de solução proposto. A seção (1.4) apresenta uma lista com as principais contribuições fornecidas pela pesquisa. A seção (1.5) descreve a organização da tese.

1.1 Alocação de Autoestruturas

A necessidade de realizar a alocação de toda a autoestrutura (autovalores, relacionados com a estabilidade e taxa de decaimento ou crescimento da resposta do sistema dinâmico, e autovetores, relacionados com a sua forma) foi observada quando verificou-se que somente a alocação de autovalores não é suficiente para assegurar a robustez de malha fechada e os requisitos de desempenho para sistemas multivariáveis, porque só a alocação de pólos não é suficiente para definir uma estrutura única de realimentação para o sistema;

isto acontece devido à liberdade promovida pela realimentação de estado ou saída em relação à seleção dos autovetores associados provenientes dos espaços disponíveis, isto é, só alocação de autovalores não é suficiente para utilizar a liberdade total de projeto. Já para o caso de sistemas com uma única entrada e uma única saída, sistemas monovariáveis, os autovalores de malha fechada definem uma estrutura única de realimentação, principalmente quando os autovalores têm liberdade para deslocar-se livremente dentro de uma certa área do plano complexo, (Liu e Patton, 1998).

A alocação de autoestruturas tem origem nos estudos dos fenômenos observados em cordas oscilantes e da condução do som pelo ar, em que verificou-se a necessidade de lidar com vários modos e frequências que ocorrem nestes sistemas. A viabilização da compreensão dos citados fenômenos ocorreu com a aplicação de métodos baseados em técnicas Lagrangeanas e Hamiltonianas que produziram o conceito de coordenadas generalizadas com suas formas e frequências modais associadas, (White, 1991); isto conduziu ao conceito de que a resposta de qualquer sistema oscilante pode ser decomposta em um conjunto de formas e frequências modais.

A alocação dos autovalores e autovetores de sistemas dinâmicos é chamada de problema de alocação de autoestruturas. Os próximos parágrafos abordam de forma breve o desenvolvimento de teorias e de métodos para alocação de autovalores e autoestruturas.

Segundo (Kailath, 1980) a alocação de autovalores foi a primeira aplicação do método de espaço de estado em sistema lineares. Talvez *J. Bertram* em 1959, segundo (Kalman *et al.*, 1969), tenha sido o primeiro pesquisador a perceber o problema da alocação de pólos e sua importância. Os trabalhos de *Kalman* certamente constituem contribuições importantes e/ou precursoras para a concepção e a solução do problema de alocação de autovalores tanto com otimização como sem otimização, (Kalman, 1960*a*) e (Kalman, 1960*b*), e inspiraram o desenvolvimento dos primeiros resultados analíticos por (Rissanen, 1960) que escreveu a primeira publicação sobre o assunto. Um trabalho de grande relevância foi desenvolvido por (Wonham, 1967), estabelecendo que os autovalores de malha fechada de qualquer sistema controlável podem ser arbitrariamente alocados através de realimentação de estado. A partir deste estudo uma série de métodos foram desenvolvidos visando a alocação dos autovalores, (Porter, 1969*a*), (Porter, 1969*b*), (Nichols e Van Dooren, 1984), (Fletcher, 1981*a*) e (Fletcher, 1981*b*), dentre outros.

Pode-se considerar os resultados apresentados por (Moore, 1976) como

muito importantes no estudo da problemática da alocação completa da autoestrutura; ele propõe um reconhecimento da liberdade oferecida, além das especificações de projeto, pela realimentação de estado em sistemas multivariáveis quando os autovalores são distintos; esta liberdade refere-se a conjuntos de autovetores. As teorias, algoritmos e aplicações desenvolvidas por (Porter e D'Azzo, 1977), (Sobel e Shapiro, 1985*a*), (Sobel e Shapiro, 1985*b*) e (Fahmy e O'Reilly, 1982) também podem ser consideradas como marcos para o estudo de alocação de autoestruturas nas últimas três décadas.

1.2 A Computação Evolutiva e os Problemas de Controle

O termo computação evolutiva, criado em 1991, representa o esforço para unir diferentes abordagens desenvolvidas para simular vários aspectos do processo da evolução natural. Reprodução, variação aleatória, competição e seleção dos indivíduos em uma população constituem a essência da evolução e são os pontos comuns das técnicas desenvolvidas, tais como: algoritmos genéticos, estratégias evolutivas e programação evolutiva. A criação destas técnicas foi estimulada pela necessidade de solucionar problemas relacionados com otimização, adaptação robusta, inteligência de máquina e compreender sistemas biológicos naturais, (Bäck *et al.*, 1997)

Os autores (Chipperfield e Fleming, 1996) optam pelo termo algoritmos evolucionistas para referenciar as abordagens que simulam o processo de evolução natural e afirmam que estes métodos têm por base modelos computacionais do processo fundamental de evolução. Estes classificam os algoritmos evolucionistas como métodos de busca estocástica e otimização.

O procedimento canônico de computação evolutiva é constituído de duas fases. A primeira fase consiste em gerar uma população inicial (solução) e realizar sua avaliação. A segunda fase consiste em evoluir a população gerada na primeira fase através da seleção de indivíduos, da criação de novos indivíduos e da avaliação deste novos indivíduos; o processo de evolução continua até certos indivíduos atingirem um nível de adequabilidade aceitável ou a população mostrar-se inapta em adaptar-se ao ambiente proposto. O procedimento a seguir ilustra o procedimento canônico:

A habilidade que os métodos de computação evolutiva possuem para realizar busca no espaço de solução, com maior probabilidade de encontrar um

ótimo global do que os métodos convencionais, torna esta abordagem bastante atrativa para resolver problemas de controle. O fato de não requerer necessariamente informações provenientes de derivadas, da estimativa inicial formal da região de solução e a natureza estocástica do mecanismo de busca justificam sua aplicação quando certos aspectos do problema são desconhecidos ou de difícil avaliação pelo projetista.

O projeto e a análise *off-line* e a adaptação e o ajuste *on-line* são classificados, de forma geral, como as duas áreas para aplicação da computação evolutiva na solução de problemas de engenharia de controle, (Chipperfield e Fleming, 1996). A citada referência comenta uma série de aplicações da computação evolutiva onde os métodos convencionais mostraram-se inadequados, problemáticos ou inexistentes; os autores ainda apresentam aplicações nos seguintes tópicos: projeto de sistemas de controle, otimização multiobjetivo, controle robusto, identificação de sistema, integração de sistema, controle adaptativo e em tempo real.

1.3 Motivação

A aplicação de técnicas de computação evolutiva para solucionar o problema de alocação de autoestruturas de sistemas dinâmicos através do projeto do regulador linear quadrático foi a principal motivação para o desenvolvimento desta pesquisa.

O parágrafo anterior apresenta a motivação que conduziu ao desenvolvimento desta pesquisa sem salientar os elos existentes entre os três elementos básicos em estudo, que são: o problema de alocação de autoestruturas, as técnicas de computação evolutiva e o projeto RLQ. A seguinte forma de pensamento conecta estes três elementos: a técnica de computação evolutiva determina as matrizes de ponderação do projeto RLQ e os ganhos do controlador fornecidos pelo projeto linear quadrático impõem a autoestrutura especificada pelo projetista ao sistema dinâmico.

1.4 Contribuições

As principais contribuições propostas nesta tese são:

- Uma formulação do problema de alocação de autoestruturas (AAE) de sistemas dinâmicos como problema de controle ótimo RLQ e de

otimização multiobjetivo.

- Um algoritmo genético paralelo para solução do problema AAE tendo como características pequena população, operador visitante e *overhead* de comunicação tendendo para zero.
- Uma unidade de decisão para direcionar a busca do algoritmo genético tendo por base o teorema do *schemata*, o paradigma do *multiarmed bandit*, a indução de nichos e o ajuste de parâmetros da operação *crossover*.
- O desenvolvimento de um sistema inteligente como ferramenta para alocação paralela de autoestruturas.

1.5 Organização da Tese

Excluindo o primeiro capítulo, onde apresenta-se generalidades sobre o problema de alocação de autoestruturas, a importância da computação evolutiva na solução de problemas de controle, a motivação desta pesquisa e as suas principais contribuições, esta tese é composta de mais seis capítulos, apêndices e índices remissivos de autores e palavras chave.

Os próximos capítulos (2-7) são organizados tendo por objetivo salientar as contribuições propostas nas áreas de controle e de computação evolutiva, mostrando sequencialmente desde a formulação do problema, as ferramentas desenvolvidas para sua solução e os resultados das aplicações da formulação e dos métodos de solução. Os capítulos (2-5) apresentam também os alicerces teóricos, seja de forma explícita ou referenciada, sobre os quais as contribuições propostas estão fundamentadas. O capítulo (6) apresenta os resultados de simulações computacionais que mostram a atuação das propostas apresentadas nos capítulos anteriormente citados. A seguir descreve-se o conteúdo dos capítulos e dos apêndices.

O capítulo (2) apresenta o problema da alocação de autoestruturas formulado como problema do regulador linear quadrático e na forma de otimização multiobjetivo. Inicialmente, discute-se o que é o problema da alocação de autoestruturas e a influência desta alocação na resposta dinâmica do sistema. O restante do capítulo apresenta o desenvolvimento da formulação proposta sempre considerando as matrizes de ponderação do problema RLQ como variáveis independentes, mostrando a formulação genérica do problema e em nível de desigualdades. A título de contextualização e revisão bibliográfica,

apresenta-se o problema *RLQ* e salienta-se a importância da problemática relacionada com a dificuldade em determinar as matrizes de ponderação que satisfaçam as condições de projeto. No intuito de mostrar a relevância da otimização multiobjetivo, apresenta-se uma sucinta revisão bibliográfica deste tema.

O capítulo (3) apresenta o algoritmo genético proposto que é chamado de otimizador genético. Os modelos das matrizes de ponderação, das operações genéticas e do algoritmo paralelo são discutidos com propósitos de formulação matemática e de ressaltar suas principais características.

O capítulo (4) apresenta os conceitos básicos e modelos que regem a unidade de decisão. A unidade de decisão é formulada com base em estratégias que são classificadas em dois tipos, segundo a sua forma de atuação, para interagir com o otimizador genético. O primeiro tipo, restringe-se a atuação sobre as operações genéticas, especificamente com as operações de *crossover*, que são classificadas de duas maneiras: a primeira trata de operações entre um indivíduo da população permanente, selecionado segundo o princípio da seleção natural de *Darwin*, e outro indivíduo, selecionado aleatoriamente e armazenado em um banco de dados de acordo com medidas de adequabilidade; a segunda maneira trata de ajustes sobre os parâmetros que regulam o grau de combinação entre dois indivíduos da operação *crossover*. O segundo tipo, restringe-se à aplicação de critérios para selecionar os membros para as populações permanentes a cada iteração do ciclo de busca e tem por base as medidas de *fitness*.

O capítulo (5) apresenta os algoritmos que formam a plataforma paralela para alocação de autoestruturas, chamada *Alocação Paralela de Autoestrutura via Algoritmo Genético Multiobjetivo e projeto RLQ*, cujo acrônimo é *APAE-AGMO/RLQ*. Esta plataforma foi construída através da formulação proposta para solução do problema de autoestrutura apresentada no capítulo (2), do algoritmo genético utilizando os modelos propostos no capítulo (3) e de uma unidade de decisão com as estratégias propostas no capítulo (4).

O capítulo (6) apresenta os resultados obtidos com a implementação da plataforma paralela. Os resultados mostram o desempenho do algoritmo, as autoestruturas alocadas e o desempenho do sistema dinâmico.

O capítulo (7) apresenta conclusões, comentários e propostas de trabalhos futuros.

Os apêndices apresentam a solução do problema *RLQ*, os ambientes computacionais utilizados, aspectos das bibliotecas de computação paralela e resultados de simulações.

Capítulo 2

Formulação do Problema

2.1 Introdução

O principal objetivo deste capítulo é apresentar o problema de alocação de autoestruturas formulado como um problema de otimização linear quadrática, especificamente o regulador linear quadrático em regime permanente, e de otimização multiobjetivo. Esta formulação constitui uma contribuição desta pesquisa para o problema de alocação de autoestruturas.

As seções do capítulo, precedentes à seção da contribuição, apresentam os fundamentos teóricos para a formulação proposta.

A seção (2.2) apresenta o problema da alocação de autoestruturas, abordando os seguintes tópicos: a necessidade de realizar este tipo de alocação, um breve histórico sobre a origem do problema, uma formulação matemática genérica do problema em nível de objetivos a serem atingidos e uma análise da influência da autoestrutura na resposta dos sistemas dinâmicos.

O problema do controle ótimo, especificamente o problema do regulador linear quadrático, sob dois aspectos é uma das peças fundamentais no desenvolvimento desta tese. O primeiro aspecto, trata da qualidade das leis de controle obtidas para a alocação da autoestrutura; todos os controladores possuem propriedades de robustez que são intrínsecas ao próprio método; o segundo aspecto, trata da justificativa de não só utilizar a teoria de algoritmos genéticos, como também contribuir para esta teoria com o desenvolvimento de um algoritmo dedicado à solução do problema proposto, porque um *gargalo* para utilização do método do regulador linear quadrático (RLQ) consiste na dificuldade que se tem em determinar as matrizes de ponderação que con-

duzam à alocação de autovalores e autovetores especificados pelo projetista. Então, devido à importância deste tópico descreve-se, de forma sucinta, na seção (2.3), o problema do controle ótimo do ponto de vista histórico, o desenvolvimento da formulação deste problema até chegar ao problema *RLQ* e a problemática da determinação das matrizes de ponderação, no intuito de salientar suas propriedades e importância dentro do projeto de sistemas de controle.

A solução proposta para o problema de alocação de autoestruturas envolve otimização multiobjetivo e método das desigualdades para determinação dos controladores que possuem a habilidade de alocar os autovalores e seus autovetores associados. A seção (2.4) apresenta a importância e uma formulação básica do problema de otimização multiobjetivo na solução de problemas de controle; a idéia desta seção é apresentar uma abstração de como a formulação de otimização multiobjetivo é utilizada para resolver o problema em questão. A seção (2.5) apresenta o método das desigualdades para o projeto de sistemas dinâmicos formulado em termos das matrizes de ponderação do problema *RLQ* e salienta as diferenças existentes entre as técnicas de projeto e as técnicas de otimização.

A seção (2.6) apresenta o problema de alocação de autoestruturas formulado como um problema de otimização multiobjetivo, método das desigualdades e problema *RLQ*. O problema *RLQ* fornece conjuntos de controladores através da seleção de suas matrizes de ponderação; estes controladores devem minimizar uma função de custo e satisfazer as restrições de projeto (limites e sensibilidades dos autovalores).

2.2 O Problema de Alocação de Autoestruturas

O problema de alocação de autoestruturas (*AAE*) tem sua origem no problema de alocação de autovalores de sistemas dinâmicos multivariáveis, em que observou-se que, para um determinado autovalor alocado, existe um conjunto de autovetores que corresponde ao autovalor especificado, o que não acontece nos sistemas monovariáveis. Devido à influência que os autovetores exercem na resposta transitória de sistemas dinâmicos, subseção (2.2.1), o problema da alocação de autoestruturas tornou-se relevante no projeto e na análise de sistemas de controle.

A possibilidade de escolher entre vários conjuntos de autovetores que satisfazem um conjunto de autovalores, um conjunto de autovetores que correspondem aos autovalores especificados, é uma liberdade que o projetista tem além da especificação dos autovalores. Esta liberdade foi inicialmente observada por (Wonham, 1967), (Kimura, 1975) e (Moore, 1976). Contudo, *Moore* foi o primeiro a caracterizar esta liberdade oferecida pela realimentação de estado de sistemas multivariáveis para autovalores distintos.

Esta seção tem por objetivo apresentar o principal resultado obtido por (Moore, 1976), salientando a caracterização da classe de todos os autovetores que podem ser obtidos para um dado conjunto de autovalores de malha fechada. Tem, também, o objetivo de apresentar um algoritmo para calcular a matriz de ganhos de malha fechada que fornece a autoestrutura desejada e que é utilizado como base para o desenvolvimento de vários métodos de alocação, como pode ser visto em (White, 1991).

A caracterização da classe de autovetores e o algoritmo sugerido são apresentados na prova da proposição de (Moore, 1976); nesta tese a prova é apresentada de forma estendida. O principal objetivo é mostrar que, se certas condições de suficiência e necessidade forem satisfeitas, existe uma matriz única K de malha fechada que impõe a autoestrutura especificada.

Quando a lei de controle $u \in R^m$ para realimentação de estado dada por:

$$u = Kx \tag{2.1}$$

onde $K^{m \times n}$ é a matriz de ganhos de realimentação e $x \in R^n$ é o vetor de estado, é aplicada no modelo linear do sistema dinâmico em variáveis de estado:

$$\dot{x} = Ax + Bu \tag{2.2}$$

$$y = Cx \tag{2.3}$$

onde, $A \in R^{n \times n}$, $B \in R^{n \times m}$, $C \in R^{r \times m}$ e $y \in R^r$ é vetor de saída, obtém-se a equação de estado linear em malha fechada:

$$\dot{x} = (A + BK)x \tag{2.4}$$

O par (A, B) deve ser controlável a fim de garantir que exista K que imponha qualquer conjunto de autovalores ao sistema. Para o sistema ser controlável, a seguinte condição deve ser satisfeita:

$$posto[B \;\; AB \ldots A^{n-1}B] = n \tag{2.5}$$

A fim de demonstrar a proposição, a seguir, considere que $\mathcal{C}$ representa o corpo do números complexos e $\mathcal{R}$ representa o corpo dos números reais.

Proposição (Moore, 1976) - Seja $\{\lambda_i\}_{i=1}^n$ um conjunto de autovalores auto-conjugados. Então, existe uma matriz $K^{m\times n}$ com seus elementos $\in \mathcal{R}$ tal que:

$$(A + BK)v_i = \lambda_i v_i, \quad 1 \leq i \leq n \tag{2.6}$$

se e somente se, para cada i, as três condições a seguir são satisfeitas:

1) Os vetores $\{v_i\}_{i=1}^n$ são linearmente independentes em $\mathcal{C}^n$,

2) $v_i = v_j^*$ sempre que $\lambda_i = \lambda_j^*$,

3) $v_i \in sub_{ger}(N^{\lambda_i})$, onde $sub_{ger}(N^{\lambda_i})$ representa o subespaço de $\mathcal{C}^n$ gerado pelas colunas de N^{λ_i}.

Se K existe e posto(B)=m, então K é única.

Prova:

Antes de iniciar a demonstração, consideram-se as seguintes definições.

Os autovalores $\lambda \in \mathcal{R}$ ($\mathcal{C}$) de malha-fechada estão associados com a matriz:

$$S^{\lambda} \triangleq \left[\begin{array}{cc} (\lambda_i - A) & B \end{array} \right] \tag{2.7}$$

onde λ_i é o *i-ésimo* elemento do conjunto de autovalores $\{\lambda_i\}_{i=1}^n$; e associada a (2.7) tem-se a matriz particionada:

$$R^{\lambda} \triangleq \left[\begin{array}{c} N^{\lambda} \\ -M^{\lambda} \end{array} \right] \tag{2.8}$$

onde as colunas de R^{λ} formam uma base para o espaço nulo de S^{λ} que é representado por: $\mathcal{N}(S^{\lambda})$. Considera-se que B tenha colunas linearmente independentes; desta forma as colunas de N^{λ} são linearmente independentes.

Se: Apresenta-se a prova da condição 3); para as outras duas condições as provas podem ser encontradas em (Bronson, 1989), (Kailath, 1980) e (Chen, 1984).

O conjunto $\{v_i\}_{i=1}^n$ é escolhido de forma a satisfazer as três condições. A partir da condição 3) monta-se um possível vetor $v_{i,1,1}$ que corresponde ao autovalor λ_i a partir de uma possível base $N^{\lambda_{i,1}}$. Então,

$$v_{i,1,1} = \begin{bmatrix} N_{1,1}^{\lambda_{i,1}} \\ N_{2,1}^{\lambda_{i,1}} \\ \dots \\ N_{n,1}^{\lambda_{i,1}} \end{bmatrix} u_{i,1}^{1} + \begin{bmatrix} N_{1,2}^{\lambda_{i,1}} \\ N_{2,2}^{\lambda_{i,1}} \\ \dots \\ N_{2,m}^{\lambda_{i,1}} \end{bmatrix} u_{i,1}^{2} + \ldots + \begin{bmatrix} N_{1,m}^{\lambda_{i,1}} \\ N_{2,m}^{\lambda_{i,1}} \\ \dots \\ N_{n,m}^{\lambda_{i,1}} \end{bmatrix} u_{i,1}^{m} \qquad (2.9)$$

onde $\begin{bmatrix} N_{1,1}^{\lambda_{i,1}} & N_{2,1}^{\lambda_{i,1}} & \dots & N_{n,1}^{\lambda_{i,1}} \end{bmatrix}'$ e $\begin{bmatrix} N_{1,m}^{\lambda_{i,1}} & N_{2,m}^{\lambda_{i,1}} & \dots & N_{n,m}^{\lambda_{i,1}} \end{bmatrix}'$ representam o primeiro e o *m-ésimo* vetores-coluna da primeira base $N^{\lambda_{i,1}}$ que geram um possível autovetor $v_{i,1,1}$ correspondente ao autovalor λ_i. $u_{i,1}^{m}$ é o *m-ésimo* componente do primeiro vetor de pesos que contribui na montagem do primeiro autovetor $v_{i,1,1}$.

Na forma matricial, tem-se para (2.9):

$$v_{i,1,1} = \begin{bmatrix} N_{1,1}^{\lambda_{i,1}} & N_{1,2}^{\lambda_{i,1}} & \dots & N_{n,m}^{\lambda_{i,1}} \\ N_{2,1}^{\lambda_{i,1}} & N_{2,2}^{\lambda_{i,1}} & \dots & N_{2,m}^{\lambda_{i,1}} \\ \dots & \dots & \dots & \dots \\ N_{n,1}^{\lambda_{i,1}} & N_{n,2}^{\lambda_{i,1}} & \dots & N_{n,m}^{\lambda_{i,1}} \end{bmatrix} \cdot \begin{bmatrix} u_{i,1}^{1} \\ u_{i,1}^{2} \\ \dots \\ u_{i,1}^{m} \end{bmatrix} \qquad (2.10)$$

Na forma matricial compacta:

$$v_{i,1,1} = \left[N_{ii,jj}^{\lambda_{i,1}}\right]\left[u_{i,1}^{jj}\right], \qquad jj = 1, \ldots, m; \quad ii = 1, \ldots, n \qquad (2.11)$$

Generalizando a equação (2.11) para um conjunto de autovetores que podem ser obtidos variando o vetor de pesos $u_{i,k}^{jj}$ e utilizando a mesma base $N_{ii,jj}^{\lambda_{i,1}}$, têm-se:

$$v_{i,k,1} = \left[N_{ii,jj}^{\lambda_{i,1}}\right]\left[u_{i,k}^{jj}\right], \quad jj = 1, \ldots, m; \quad ii = 1, \ldots n; \quad k = 1, \ldots, k_p \qquad (2.12)$$

onde k_p é a quantidade de autovetores $v_{i,k,1}$ que correspondem ao *i-ésimo* autovalor e que são obtidos a partir da base $N_{ii,jj}^{\lambda_{i,1}}$ e do conjunto de vetores de peso $u_{i,k}^{jj}$.

A última generalização é feita considerando a formação de autovetores $v_{i,k}$ a partir de quaisquer conjuntos de vetores linearmente independentes que constituem a base $N_{ii,jj}^{\lambda_{i,l}}$. Então,

$$v_{i,k,l} = \left[N_{ii,jj}^{\lambda_{i,l}}\right]\left[u_{i,k}^{jj}\right], \quad \begin{matrix} jj = 1,\ldots m, & ii = 1,\ldots n \\ k = 1,\ldots,k_p, & l = 1,\ldots,l_b \end{matrix} \tag{2.13}$$

onde $v_{i,k,l}$ representa o *i-ésimo* autovetor, que é gerado por um *k-ésimo* conjunto de vetores de pesos $u_{i,k}^{jj}$ e uma *l-ésima* base $N_{ii,jj}^{\lambda_{i,l}}$, correspondente ao *i-ésimo* autovalor λ_i. l_b representa a quantidade de bases que gera um conjunto de autovetores que correspondem ao *i-ésimo* autovalor.

Com isto, a partir da condição 3) mostrou-se, que para um dado conjunto de autovalores distintos, existe a possibilidade de associar vários conjuntos de autovetores. Por uma questão de simplicidade, (2.13) é representada por:

$$v_i = N^{\lambda_i} u_i \qquad i = 1,\ldots n \tag{2.14}$$

onde $v_i = v_{i,k,l}$, $N^{\lambda_i} = \left[N_{ii,jj}^{\lambda_{i,l}}\right]$ e $u_i = \left[u_{i,k}^{jj}\right]$.

Considere que K é escolhido tal que $(A + BK)$ tenha um autovalor desejado λ_i com o correspondente autovetor v_i:

$$(A + BK)\, v_i = v_i \lambda_i \tag{2.15}$$

portanto,

$$(\lambda_i I - A)\, v_i - BKv_i = 0 \tag{2.16}$$

Na forma matricial:

$$\left[\begin{array}{cc} \lambda_i I - A & B \end{array}\right] \left[\begin{array}{c} v_i \\ -Kv_i \end{array}\right] = 0 \tag{2.17}$$

No intuito de determinar um K que aloque λ_i como um autovalor da malha fechada, deve-se primeiramente determinar uma base $R^\lambda = \left[\begin{array}{cc} N^\lambda & -M^\lambda \end{array}\right]'$ para o espaço nulo da matriz $S_{n\times(n+m)}^{\lambda}$ que é um subespaço do $\mathcal{R}^{n+m}$ definido na forma:

$$\mathcal{N}(S^\lambda) = \left\{R^\lambda u_i \in \mathcal{R}^{n+m} : S^\lambda R^\lambda u_i = 0\right\} \tag{2.18}$$

Como $R^{\lambda} = \left[\begin{array}{cc} N^{\lambda} & -M^{\lambda} \end{array} \right]'$, de (2.7) e da definição de espaço nulo:

$$\left[\begin{array}{cc} (\lambda_i I - A) & B \end{array} \right] . \left[\begin{array}{c} N^{\lambda} u_i \\ -M^{\lambda} u_i \end{array} \right] = 0 \tag{2.19}$$

verifica-se que a dimensão desta base é $(n+m) - posto \left[\begin{array}{cc} \lambda_i I - A & B \end{array} \right] = (n+m) - n = m$ onde posto $\left[\begin{array}{cc} \lambda_i I - A & B \end{array} \right]$= n, pois o par (A, B) é controlável. Então, como R^{λ} é uma base, existe um vetor de pesos $u_i \neq 0$, $m \times 1$, tal que:

$$\left[\begin{array}{c} N^{\lambda_i} \\ -M^{\lambda_i} \end{array} \right] u_i = \left[\begin{array}{c} v_i \\ -K v_i \end{array} \right] \tag{2.20}$$

Considerando as relações:

$$M^{\lambda_i} u_i = K v_i \tag{2.21}$$

$$N^{\lambda_i} u_i = v_i \tag{2.22}$$

tem-se

$$K N^{\lambda_i} u_i = M^{\lambda_i} u_i \tag{2.23}$$

K deve satisfazer (2.23) para que λ_i seja um autovalor da malha fechada. O vetor u_i, pode ser escolhido arbitrariamente. De (2.22) note que $N^{\lambda_i} u_i = v_i$ é o autovetor correspondente a λ_i e que u_i representa a flexibilidade que se tem para selecionar o autovetor correspondente, em adição à alocação de um autovalor. Portanto, se o autovetor v_i, $n \times 1$, não pode ser arbitrariamente alocado, por outro lado o vetor u_i, $m \times 1$, pode em geral ser arbitrariamente selecionado.

Somente Se: Supondo (2.23) tem-se

$$[\lambda_i I - (A + BK)] N^{\lambda_i} u_i = (\lambda_i I - A) N^{\lambda_i} u_i - B M^{\lambda_i} u_i \tag{2.24}$$

donde, de (2.19):

$$[\lambda_i I - (A + BK)] N^{\lambda_i} u_i = 0 \tag{2.25}$$

Portanto λ_i é um autovalor de $A + BK$ e $N^{\lambda_i} u_i$ é o autovetor correspondente. A equação (2.25) mostra que o autovetor v_i pertence ao espaço

nulo de $[\lambda_i I - (A + BK)]$ e corresponde ao *i-ésimo* autovalor. Por sua vez este autovetor é gerado através de uma base N^{λ_i}. Mostrou-se que para cada *i-ésimo* autovalor pode-se associar um conjunto de autovetores que é gerado através da combinação linear dos vetores coluna da base N^{λ_i}, ou seja, $v_i \in sub_{ger}(N^{\lambda_i})$.

A prova é finalizada quando mostra-se que uma matriz de número reais $K^{m \times n}$, que impõe a autoestrutura, pode ser construída a partir dos vetores que pertencem ao espaço nulo de $\begin{bmatrix} (A - \lambda_i I) & B \end{bmatrix}$, que são os autovetores v_i e os vetores $\begin{bmatrix} M^{\lambda_i} \end{bmatrix} u_i$.

A matriz de ganhos K, que impõe a autoestrutura especificada, é determinada quando escreve-se (2.23) para o conjunto de n autovalores especificados $\{\lambda_i\}_{i=1}^{n}$ e seleciona-se os vetores de peso u_i tal que os autovetores correspondentes $v_i = N^{\lambda_i} u_i$ sejam linearmente independentes. Então,

$$KV = S \tag{2.26}$$

onde, $V = \begin{bmatrix} N^{\lambda_1} u_1 & N^{\lambda_2} u_2 & \dots & N^{\lambda_n} u_n \end{bmatrix}$ e $S = \begin{bmatrix} M^{\lambda_1} u_1 & M^{\lambda_2} u_2 \dots & M^{\lambda_n} u_n \end{bmatrix}$.

Manipulando (2.26), isto é: exprimindo esta relação em termos de uma matriz $n \times n$ formada pelos n autovetores associados com os n autovalores λ_i e de uma matriz $m \times n$ formada pelos vetores $M^{\lambda_i} u_i$, $1 \leq i \leq n$, determina-se a matriz K:

$$K = \begin{bmatrix} M^{\lambda_1} u_1 & M^{\lambda_2} u_2 & \dots & M^{\lambda_n} u_n \end{bmatrix} . \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}^{-1} \tag{2.27}$$

onde $V = \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}$.

Quando os λ_i são distintos, sempre é possível selecionar u_i tal que V tenha posto completo; de fato, qualquer $u_i \neq 0$ é suficiente. Quando λ_i tem valores repetidos pode ser ainda possível, sob certas condições selecionar, u_i tal que $N^{\lambda_i} u_i$, $i = 1, \dots, n$, sejam linearmente independentes. Contudo, em geral, para autovalores repetidos a equação (2.26) precisa ser modificada e detalhes podem ser encontrados em (Moore, 1976) e (Klein e Moore, 1976).

Quando os autovalores são complexos conjugados, aplica-se uma transformação T_{CR} para que os elementos da matriz de autovetores e da matriz $[M^{\lambda_1} u_1 \quad M^{\lambda_2} u_2 \quad \dots \quad M^{\lambda_n} u_n]$ sejam reais. Então, para o caso de autovalores complexos:

$$K\begin{bmatrix} \dots & v_{iR}+jv_{iI} & v_{iR}-jv_{iI} & \dots \end{bmatrix} = \begin{bmatrix} \dots & s_{iR}+js_{iI} & s_{iR}-js_{iI} & \dots \end{bmatrix} \tag{2.28}$$

a transformação é dada por:

$$T_{CR} = \begin{bmatrix} I & | & & & | & \\ -- & -+- & -- & -- & -+- & -- \\ & | & \frac{1}{2} & -j\frac{1}{2} & | & \\ & | & \frac{1}{2} & j\frac{1}{2} & | & \\ -- & -+- & -- & -- & -+- & -- \\ & | & & & | & I \end{bmatrix} \tag{2.29}$$

Aplicando a transformação T_{CR} em (2.28) obtém-se a equação (2.30) apenas com elementos reais:

$$K\begin{bmatrix} \dots & v_{iR} & v_{iI} & \dots \end{bmatrix} = \begin{bmatrix} \dots & s_{iR} & s_{iI} & \dots \end{bmatrix} \tag{2.30}$$

Porque as n colunas da matriz de autovetores V são linearmente independentes, e a matriz $\begin{bmatrix} \dots & v_{iR} & v_{iI} & \dots \end{bmatrix}$ possui posto n.

A relação (2.26) mostra que o K que aloca todos os autovalores de malha fechada não é único. Se o par (A, B) é controlável, considere que n autovalores foram escolhidos para $A + BK$. A matriz de realimentação de estado K que aloca todos os autovalores nos locais especificados não é única para o caso multivariável ($m > 1$), ver (Chen, 1984), (Antsaklis e Michel, 1997) e (Fartes Filho, 1977), contudo ela é única para o caso monovariável ($m = 1$).

Desde que a matriz de malha fechada $A + BK$ seja unicamente definida pelos seus autovalores distintos e autovetores, a matriz K é única se a matriz B tiver vetores coluna linearmente independentes. Neste caso, a equação (2.26) permite, através da especificação de V e de S, determinar K de forma única como solução de um conjunto de n equações linearmente independentes.

2.2.1 A AE e a Resposta do Sistema Dinâmico

Apresenta-se nesta subseção a influência exercida pela autoestrutura (AE) na resposta temporal do sistema dinâmico. A idéia básica é expressar a equação de estado em termos da autoestrutura e, logo após, fazer uma análise da influência dos autovalores e autovetores na resposta do sistema. Esta seção

tem como referência os trabalhos de *Bottura*, (Bottura, 2000), e *Mac Farlane*, (Mac Farlane, 1974).

Seja o sistema linear homogêneo:

$$\dot{x} = Ax \tag{2.31}$$

com o conjunto de n autovalores distintos $\{\lambda_i\}_{i=1}^{n}$, aos quais correspondem n autovetores normalizados $\{v_i\}_{i=1}^{n}$ linearmente independentes que constituem uma base do espaço n dimensional e são definidos por:

$$Av_i = \lambda_i v_i, \quad \langle v_i, v_j \rangle = \delta_{i,j}, \quad i,j = 1,2,\ldots,n \tag{2.32}$$

$$\delta_{i,j} = \begin{cases} 1, & \text{Se } i = j \\ 0, & \text{Se } i \neq j \end{cases} \tag{2.33}$$

Portanto pode-se exprimir o vetor de estado x como:

$$x(t) = \sum_{i=1}^{n} a_i(t) v_i \tag{2.34}$$

Aplicando uma transformação de equivalência

$$x = Vz \tag{2.35}$$

obtém-se a forma normal ou modal

$$\dot{z} = \Lambda z = \begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & & & \\ 0 & 0 & \ldots & \lambda_n \end{bmatrix} z \tag{2.36}$$

onde $\Lambda = V^{-1}AV$ e as colunas de V, conhecida como matriz modal, são os autovetores v_i associados aos autovalores λ_i e as colunas de V^{-1} são os autovetores recíprocos ou autovetores à esquerda.

Ainda,

$$\dot{z}_i = \lambda_i z_i; \quad i = 1,2,\ldots,n \tag{2.37}$$

cujas soluções são imediatas

$$z_i = \alpha_i e^{\lambda_i t}; \quad i = 1, 2, \ldots, n; \quad \alpha_i = z_i(0) \tag{2.38}$$

Da equação (2.38), verifica-se de imediato que:

1. O sistema será assintoticamente estável se todos os autovalores tiverem parte real negativa.

2. Se certos autovalores tiverem parte real negativa de magnitude bem maior do que as outras, suas contribuições transitórias serão desprezíveis e poderemos tratar o sistema como tendo uma ordem inferior a n.

Portanto,

$$x = Vz = \begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix} \begin{bmatrix} \alpha_1 e^{\lambda_1 t} \\ \alpha_2 e^{\lambda_2 t} \\ \ldots \\ \alpha_n e^{\lambda_n t} \end{bmatrix} = \sum_{i=1}^{n} \alpha_i e^{\lambda_i t} v_i \tag{2.39}$$

e de (2.34) e (2.39)

$$x = \sum_{i=1}^{n} a_i(t) v_i = \sum_{i=1}^{n} \alpha_i e^{\lambda_i t} v_i \tag{2.40}$$

que mostra que para cada vetor característico v_i corresponde um modo $e^{\lambda_i t} v_i$ e que a resposta livre é uma combinação linear dos modos do sistema.

As constantes podem ser facilmente determinadas pelo emprego da base recíproca.

Se $\begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix}$ constituem uma base de um espaço χ de dimensão n, o conjunto de vetores $\begin{bmatrix} w_1 & w_2 & \ldots & w_n \end{bmatrix}$ definidos tal que

$$\langle v_i, w_j \rangle = \delta_{i,j}, \quad i, j = 1, 2, \ldots, n \tag{2.41}$$

é a base recíproca do espaço χ.

De (2.40), para autovalores reais:

$$x(0) = \sum_{i=1}^{n} \alpha_i v_i \tag{2.42}$$

Fazendo o produto escalar de (2.42) com w_i obtém-se:

$$\langle w_i, x(0)\rangle = \alpha_1\langle w_i, v_1\rangle + \ldots + \alpha_i\langle w_i, v_i\rangle + \ldots + \alpha_n\langle w_i, v_n\rangle = \alpha_i \quad (2.43)$$

donde

$$\alpha_i = \langle w_i, x(0)\rangle \quad (2.44)$$

e escreve-se a resposta livre do sistema como:

$$x(t) = \sum_{i=1}^{n} \langle w_i, x(0)\rangle e^{\lambda_i t} v_i \quad (2.45)$$

A equação (2.44) representa a magnitude da excitação do *i-ésimo* modo devido às condições iniciais. Para autovalores reais, se as condições iniciais forem tais que $\alpha_i = 0$ para $i \neq j$, ou seja, se apenas o *j-ésimo* autovetor for excitado, então,

$$x(t) = \alpha_j e^{\lambda_j t} v_j \quad (2.46)$$

Manipulando (2.31) e derivando (2.46)

$$\dot{x}(t) = Ax(t) = \alpha_j e^{\lambda_j t} A v_j = \alpha_j \lambda_j e^{\lambda_j t} v_j \quad (2.47)$$

Portanto, neste caso, o vetor velocidade instantânea $\dot{x}(t)$, tangente à trajetória de estado, é orientado ao longo do autovetor, donde, qualquer trajetória que começa ou que intercepta um autovetor permanece sobre esta direção característica.

Além disto, tem-se que

$$\frac{\dot{x}(t)}{x(t)} = \lambda_j, \quad j = 1, 2, \ldots, n \quad (2.48)$$

Analogamente, para o sistema linear estacionário forçado,

$$\dot{x}(t) = Ax(t) + Bu(t) \quad (2.49)$$

pode-se decompor a função forçante ao longo dos autovetores v_i fazendo

$$Bu(t) = H(t) = \sum_{i=1}^{n} h_i(t) v_i \tag{2.50}$$

onde $h_i(t) = \langle w_i, H(t) \rangle = \langle w_i, Bu(t) \rangle$

Desta forma, podemos escrever a equação de transição de estado como

$$x(t) = \sum_{i=1}^{n} \langle w_i, x(0) \rangle e^{\lambda_i t} v_i + \int_0^t \sum_{i=1}^{n} \langle w_i, Bu(\tau) \rangle e^{\lambda_i (t-\tau)} v_i d\tau \tag{2.51}$$

A equação (2.51) esclarece a influência da autoestrutura na resposta do sistema dinâmico, caracterizando esta resposta como uma combinação linear dos n modos $e^{\lambda_i t} v_i$, $i = 1, \ldots, n$, onde cada autovalor λ_i e o seu correspondente autovetor à direita v_i definem a característica no domínio do tempo e a forma do *i-ésimo* modo. A referida equação também ressalta o efeito da entrada sobre cada modo.

Se a entrada for escolhida de tal maneira que apenas o *j-ésimo* modo é excitado e se o sistema for estável, teremos em regime

$$x_\tau(t) = \int_0^t \sum_{i=1}^{n} \langle w_j, Bu(\tau) \rangle e^{\lambda_i (t-\tau)} v_j d\tau \tag{2.52}$$

Estas observações são fundamentais para o projeto de testes de grandes sistemas, como por exemplo, ensaios de grandes estruturas, de sistemas complexos de controle e outros.

No caso de autovalores complexos, que para sistemas reais ocorrem em pares conjugados, donde também os correspondentes autovetores e os correspondentes da base recíproca:

$$\lambda_1 = \sigma_1 + j\varpi_1, \quad v_1 = v_{1R} + j v_{1I}, \quad w_1 = w_{1R} + j w_{1I} \tag{2.53}$$

$$\lambda_2 = \sigma_1 - j\varpi_1, \quad v_2 = v_{1R} - j v_{1I}, \quad w_2 = w_{1R} - j w_{1I} \tag{2.54}$$

As seguintes relações são válidas:

$$\langle w_{1R}, v_{1R} \rangle = 1, \quad \langle w_{1R}, v_{1I} \rangle = 0 \tag{2.55}$$

$$\langle w_{1I}, v_{1R} \rangle = 0, \quad \langle w_{1I}, v_{1I} \rangle = 1 \tag{2.56}$$

com a condição de normalização $\langle v_{1R}, v_{1R}\rangle + \langle v_{1I}, v_{1I}\rangle = 1$. Desta forma, a contribuição para a resposta livre de p pares de autovalores complexos conjugados será:

$$\begin{aligned} x(t) &= \sum_{i=1}^{p} e^{\sigma_i t}\{[\langle w_{1R}, x(0)\rangle \cos(\varpi_i t) + \langle w_{1I}, x(0)\rangle \sin(\varpi_i t)]\, v_{1R} + \\ &+ [\langle w_{1I}, x(0)\rangle \cos(\varpi_i t) - \langle w_{1R}, x(0)\rangle \sin(\varpi_i t)]\, v_{1I}\} \end{aligned} \tag{2.57}$$

A amplitude e a fase de cada modo oscilatório dependem das condições iniciais. Por exemplo, se $x(0) = v_{1R}$ apenas o primeiro modo oscilatório ou de 2ª ordem é excitado:

$$x(t) = e^{\sigma_1 t}[\cos(\varpi_1 t)v_{1R} - \sin(\varpi_i 1)v_{1I}] \tag{2.58}$$

Por outro lado, se $x(0) = v_{1I}$, excita-se apenas o 2º modo de 2ª ordem:

$$x(t) = e^{\sigma_1 t}[\sin(\varpi_1 t)v_{1R} + \cos(\varpi_i 1)v_{1I}] \tag{2.59}$$

Portanto, para um par de autovalores complexos conjugados, o movimento no espaço de estado tem a forma de uma espiral logarítmica no plano (v_{1R}, v_{1I}).

Substituindo (2.51) em (2.3), obtém-se uma expressão que permite salientar simultaneamente a influência da autoestrutura tanto em nível de entrada quanto de saída do sistema dinâmico. Então,

$$y(t) = C\sum_{i=1}^{n} v_i e^{\lambda_i t}\langle w_i, x(0)\rangle + C\int_0^t \sum_{i=1}^{n} v_i e^{\lambda_i (t-\tau)}\langle w_i, Bu(\tau)\rangle d\tau \tag{2.60}$$

Fazendo

$$\gamma_i = Cv_i, \qquad i = 1, 2, \ldots n \tag{2.61}$$

então γ_i são as colunas da matriz CV

$$\begin{bmatrix} \gamma_1 & \gamma_2 & \ldots & \gamma_n \end{bmatrix} = C \begin{bmatrix} v_1 & v_2 & \ldots & v_n \end{bmatrix} \tag{2.62}$$

e

$$\beta_j' = w_j' B, \qquad j = 1, 2, \ldots n \tag{2.63}$$

β_j' são as linhas da matriz WB

$$\begin{bmatrix} \beta_1' \\ \beta_2' \\ \cdots \\ \beta_n' \end{bmatrix} = \begin{bmatrix} w_1' \\ w_2' \\ \cdots \\ w_n' \end{bmatrix} B \tag{2.64}$$

Como o conjunto dos n autovetores $\{w_i\}_{i=1}^n$ é o recíproco do conjunto dos n autovetores $\{v_i\}_{i=1}^n$, tem-se que:

$$W = V^{-1} \tag{2.65}$$

Substituindo (2.61) e (2.63) em (2.60) e sabendo que $\langle w_i, Bu(\tau)\rangle = \langle B'w_i, u(\tau)\rangle$, tem-se

$$y(t) = \sum_{i=1}^{n} \gamma_i e^{\lambda_i t}\langle w_i, x(0)\rangle + \sum_{i=1}^{n} \int_0^t \gamma_i e^{\lambda_i (t-\tau)}\langle \beta_i, u(\tau)\rangle d\tau \tag{2.66}$$

O termo integral é uma convolução, então:

$$y(t) = \sum_{i=1}^{n} \gamma_i e^{\lambda_i t}\langle w_i, x(0)\rangle + \sum_{i=1}^{n} \gamma_i e^{\lambda_i t} * \langle \beta_i, u(t)\rangle \tag{2.67}$$

Observando a equação (2.67), verifica-se que γ_i, $\langle w_i, x(0)\rangle$ e $\langle \beta_i, u(t)\rangle$ relacionam as saídas, as condições iniciais e as entradas, respectivamente, com os autovetores, estabelecendo a magnitude modal $e^{\lambda_i t}$.

Associado ao espaço de saída há um conjunto de n vetores coluna γ_i e associado ao espaço de entrada, Figura (2.1), há um conjunto de n vetores linha β_i'.

Os vetores linha β_i' são as linhas de $WB = V^{-1}B$ e representam o acoplamento entre as entradas e os autovetores de A, que caracterizam o movimento no espaço de estado. Os vetores coluna γ_i são as colunas de CV e representam o acoplamento entre os autovetores no espaço de estado e o espaço de saída.

Todos os movimentos possíveis no espaço de estado podem ser expressos como uma soma dos movimentos característicos associados com os autovalores, os chamados modos como já visto. Uma entrada $u(t)$ no sistema é acoplada ao *i-ésimo* modo pela realização do produto escalar $\langle \beta_i, u(t)\rangle$; este

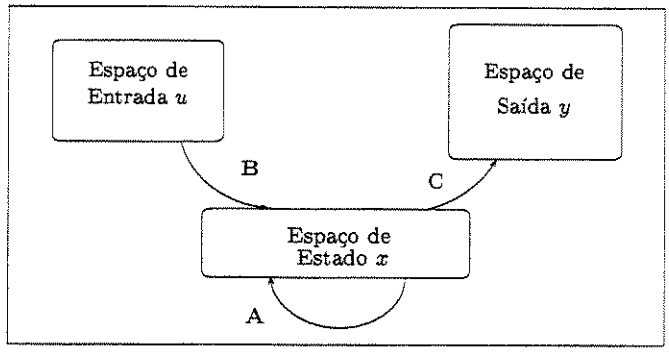

Figura 2.1: Mapeamentos do Modelo Linear em Variáveis de Estado

componente é unido com o movimento característico exponencial para obter-se:

$$e^{\lambda_i}\langle \beta_i, u(t)\rangle \tag{2.68}$$

Portanto a saída é composta da soma dos movimentos ao longo do conjunto de vetores γ_i do espaço de saída como:

$$saída = \sum_{i=1}^{n} \gamma_i e^{\lambda_i}\langle \beta_i, u(t)\rangle \tag{2.69}$$

A controlabilidade e a observabilidade dos sistemas dinâmicos podem ser avaliadas através do acoplamento dos vetores modais com as matrizes de controle B e de saída C.

Aplicando a transformada de *Laplace* em (2.2-2.3), considerando as condições iniciais nulas, e substituindo a solução da equação de estado $x(s)$ na saída, tem-se a matriz de transferência $G(s)$:

$$G(s) = C(sI - A)^{-1}B \tag{2.70}$$

Aplicando a transformada inversa de *Laplace* em (2.70), no domínio do tempo tem-se a matriz de resposta ao impulso $G(t) = \mathcal{L}^{-1}[G(s)]$:

$$G(t) = Ce^{At}B \tag{2.71}$$

De (2.36) e (2.71) tem-se

$$G(t) = Ce^{V\Lambda Wt}B \tag{2.72}$$

Expandindo em série de *Taylor* o termo $e^{V\Lambda Wt}$ e depois retornando ao termo exponencial, obtém-se este termo expresso em termos da autoestrutura da matriz A:

$$\begin{aligned} e^{V\Lambda Wt} &= I + V\Lambda tW + \frac{(V\Lambda W)^2t^2}{2!} + \frac{(V\Lambda W)^3t^3}{3!} + \ldots \\ &= V(I + \Lambda t + \frac{\Lambda^2t^2}{2!} + \frac{\Lambda^3t^3}{3!} + \ldots)W \\ &= Ve^{\Lambda t}W \end{aligned} \tag{2.73}$$

Substituindo a forma exponencial final (2.73) em (2.72), tem-se:

$$G(t) = CVe^{\Lambda t}WB \tag{2.74}$$

Considerando que todos os autovalores Λ são distintos e sabendo-se de (2.63) que $\beta_i' = w_i'B$ e de (2.61) que $\gamma_i = Cv_i$, (2.74) pode ser expressa como:

$$G(t) = \sum_{i=1}^{n} \gamma_i e^{\Lambda t} \beta_i' \tag{2.75}$$

Devido à clareza com que $G(t)$ em (2.75) salienta o acoplamento da saída com os vetores coluna γ_i e da entrada com os vetores linha β_i', esta equação é útil para mostrar o efeito dos modos na controlabilidade e na observabilidade do sistema dinâmico.

Quando vistos através da matriz de saída C, os vetores modais $\{v_i\}_{i=1}^n$ são representados pelos vetores coluna $\{\gamma_i\}_{i=1}^n$. Se qualquer vetor γ_i tende para zero, o *i-ésimo* modo é não observável, caso contrário é observável; se todos os modos são observáveis o sistema é totalmente observável. Desta maneira, verifica-se a observabilidade dos modos e do sistema através do acoplamento dos vetores modais com as linhas da matriz de saída.

De forma similar, quando vistos através da matriz de entrada B os vetores modais recíprocos $\{w_i\}_{i=1}^n$ são representados pelos vetores linha $\{\beta_i'\}_{i=1}^n$. Se qualquer vetor β_i' tende para zero, o *i-ésimo* modo é não controlável, caso contrário é controlável; se todos os modos são controláveis o sistema é totalmente controlável. Desta maneira verifica-se a controlabilidade dos modos e do sistema através do acoplamento dos vetores modais recíprocos com as colunas da matriz de entrada.

2.2.2 Uma Formulação Genérica do Problema

Esta subseção apresenta uma formulação genérica do problema de alocação de autoestruturas por realimentação de estado, estimulada pelas abordagens de *D'Azzo*, (D'Azzo e Houpis, 1995), e *Andry et al.*, (Andry Jr. *et al.*, 1983), e visa generalizar uma alocação que tem por base satisfazer restrições de autovalores e autovetores, que é bastante diferente das formulações clássicas apresentadas em (White, 1995).

Seja um sistema dinâmico linear multivariável, modelado no espaço de estado, linearizado e invariante no tempo, representado por:

$$\delta x = Ax + Bu \tag{2.76}$$

$$y = Cx \tag{2.77}$$

onde δx é $\dot{x}$ para sistemas contínuos e $x(k+1)$ para sistema discretos, $x \in R^n$ representa o vetor de estado, $u \in R^m$ representa o vetor de controle, $y \in R^r$ representa o vetor de saída. As matrizes $A \in R^{n \times n}$, $B \in R^{n \times m}$ e $C \in R^{r \times n}$ representam a dinâmica do sistema, a ponderação do controle e a saída, respectivamente.

Considera-se que o par (A, B) é controlável e que o par (C, A) é observável. O posto da matriz B é igual a m e o posto da matriz C é igual a r.

O problema de alocação de autoestruturas é estabelecido pela determinação de uma lei de controle u:

$$u = Kx \tag{2.78}$$

onde, $K^{m \times n}$ é a matriz de ganho de realimentação de estados.

Esta lei de controle é realimentada no modelo do sistema dinâmico, equação (2.76), e resulta o sistema de malha fechada:

$$\delta x = (A + BK)x \tag{2.79}$$

A autoestrutura (AE) imposta ao sistema de malha fechada, equação (2.79), pela lei de controle, equação (2.78), deve satisfazer as condições de projeto, que são as restrições de autovalores e autovetores, a fim de que resulte a autoestrutura especificada pelo projetista.

As restrições dos autovalores são especificadas por uma região no semiplano complexo esquerdo (*SPCE*) e esta região é delimitada pelas desigualdades:

$$\lambda_{i,ee} \leq \lambda_{i,c} \leq \lambda_{i,ed}, \qquad i = 1, \ldots, n \tag{2.80}$$

onde $\lambda_{i,ee}$ e $\lambda_{i,ed}$ são os *i-ésimos* valores complexos conjugados representando os limites especificados à esquerda, $Real(\lambda_{i,ee}) \pm jImag(\lambda_{i,ee})$, e à direita, $Real(\lambda_{i,ed}) \pm jImag(\lambda_{i,ed})$ do *SPCE*, respectivamente, com $Real(\lambda_{i,ed}) > Real(\lambda_{i,ee})$; $\lambda_{i,c}$ corresponde ao *i-ésimo* autovalor alocado que satisfaz as restrições $\lambda_{i,ee}$ e $\lambda_{i,ed}$.

Os dois limites complexos conjugados, $\lambda_{i,ee}$ e $\lambda_{i,ed}$, estabelecem o contorno de uma região do *SPCE*, Figura (2.2), de forma tal que qualquer autovalor dentro desta região satisfaz as restrições dos autovalores. Quatro pontos no *SPCE* são suficientes para definir as fronteiras desta região, o par de pontos obtidos através da restrição à esquerda, $\lambda_{i,ee}$, é $(Real(\lambda_{i,ee}), Imag(\lambda_{i,ee}))$ e $(Real(\lambda_{i,ee}), -Imag(\lambda_{i,ee}))$; de forma similar, o par de pontos obtidos através da restrição à direita, $\lambda_{i,ed}$, é: $(Real(\lambda_{i,ed}), Imag(\lambda_{i,ed}))$ e $(Real(\lambda_{i,ed}), -Imag(\lambda_{i,ed}))$.

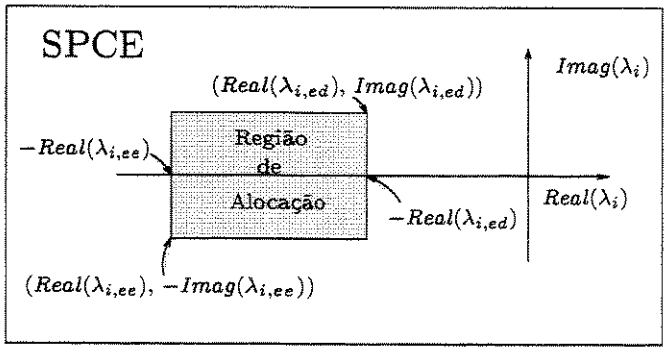

Figura 2.2: Região de Alocação para $\lambda_{i,c}$.

Um vetor de n componentes, $(\epsilon_1, \epsilon_2, \ldots, \epsilon_n)$, define as restrições dos autovetores e para cada componente está associada uma função dos autovetores à direita e à esquerda, especificados pelo projetista. Então, os i-ésimos autovetores à esquerda e à direita, $v_{i,mf}$ e $w_{i,mf}$, do sistema de malha fechada, equação (2.79), são mapeados em um conjunto de números reais através de

alguma transformação, $f_i(v_{i,mf},\ w_{i,mf})$, e a alocação de autovetores deverá satisfazer às restrições:

$$f_i(v_{i,mf}, w_{i,mf}) \leq \epsilon_i, \qquad i = 1, \ldots, n \tag{2.81}$$

onde $f_i(.)$ é a i-ésima função escalar dos autovetores alocados à direita, $v_{i,mf}$, e à esquerda, $w_{i,mf}$ que devem satisfazer à i-ésima restrição de projeto ϵ_i.

2.3 O Controle Ótimo

O problema de controle denominado regulador linear quadrático (RLQ) está inserido na teoria de controle ótimo. O problema do controle ótimo tem como opções para sua solução uma variedade de técnicas, dentre as quais destacam-se a teoria de *Hamilton-Jacobi-Bellman* e o princípio do mínimo de *Pontryagin*. No intuito de estabelecer matematicamente o problema RLQ, apresenta-se a formulação geral do problema de controle ótimo para o caso contínuo e suas propriedades, tendo por base (Athans e Falb, 1966), (Sage, 1968) e (Luenberger, 1979).

Seja um sistema dinâmico modelado em variáveis de estado,

$$\dot{x} = f(x(t), u(t)) \tag{2.82}$$

onde o vetor $x(t)$ de dimensão n representa os estados. O vetor $u(t)$ de dimensão m representa a lei de controle. Considere que o modelo está definido no intervalo fixo $0 \leq t \leq T$.

As condições iniciais $x(0)$ do sistema dinâmico, equação (2.82), são dadas por:

$$x(0) = x_0 \tag{2.83}$$

A lei de controle $u(t)$ é dada por:

$$u(t) \in U \tag{2.84}$$

onde U é o conjunto das leis de controle. Este conjunto pode ser todo o espaço R^m ou pode ser limitado por desigualdades de acordo com o tipo de aplicação em questão.

O problema de controle ótimo tem por objetivo a determinação de uma lei de controle que otimiza um índice de desempenho (*função de custo ou*

objetivo) e tenha como restrição o sistema dinâmico, equação (2.82). O índice de desempenho a ser otimizado é dado por:

$$J = \psi(x(T)) + \int_0^T l(x(t), u(t))dt \tag{2.85}$$

onde ψ e l são funções reais. O valor real $\psi(x(T))$ é a contribuição do estado final e o valor real de $\int_0^T l(x(t), u(t))dt$ é a contribuição que se acumula no intervalo $0 \leq t \leq T$.

Sob a luz das equações (2.82-2.85) tece-se os seguintes comentários. A função de custo é responsável pela caracterização do desempenho do sistema dinâmico quando utiliza-se a abordagem de controle ótimo para determinar as leis de controle que conduzam a um comportamento especificado pelo projetista. Este índice é um escalar que pode incluir de forma combinada e ponderada uma série de medidas cujos valores estão relacionados com a qualidade do desempenho do sistema; estas medidas podem ser: energia mínima para controlar o sistema dinâmico, tempo mínimo para atingir um dado estado de operação, entre outras. O valor deste índice só é conhecido após o término do processo de otimização.

Este ponto é bastante oportuno para traçar um paralelo entre o controle ótimo e o clássico. Enquanto que no controle clássico, as especificações do sistema são conhecidas a *priori* pelo projetista, tais como estabilidade relativa (margens de ganho ou fase) ou características de desempenho do sistema, isto não acontece no controle ótimo conforme mencionado anteriormente.

2.3.1 O Problema do Regulador Linear Quadrático

O problema do regulador linear quadrático tem suas origens nos trabalhos de *Wiener*, (Wiener, 1948), que utilizou uma função de desempenho erro quadrático-médio no projeto de filtros. A primeira contribuição para síntese de controladores com realimentação tem como precursor *Kalman*, (Kalman *et al.*, 1958), utilizando a teoria de controle linear quadrático. A atual formulação do problema conhecida como regulador linear quadrático foi estabelecida por *Kalman*, (Kalman, 1960*b*), e *Letov*, (Letov, 1960).

A partir dos marcos citados no parágrafo anterior, teorias e metodologias vêm sendo desenvolvidas para análise, síntese e implementação de controladores que têm por base o problema do regulador linear quadrático (*RLQ*), (Athans e Falb, 1966), (Sage, 1968), (Luenberger, 1979), (Bryson e Ho, 1975) e (Dorato *et al.*, 1995).

Partindo das referências citadas, apresenta-se no Apêndice (A) a formulação e a solução do problema RLQ em regime permanente, para o sistema dinâmico invariante no tempo e intervalo de otimização infinito. As equações de otimização do problema RLQ podem ser obtidas utilizando Programação Dinâmica ou Princípio do Máximo de *Pontryagin*, ou Cálculo Variacional. Neste trabalho, escolheu-se Programação Dinâmica para derivação das equações de otimização porque estas conduzem de forma direta à equação de *Riccati* e esta equação é parte integrante da função de *fitness* do algoritmo de otimização multiobjetivo que será proposto.

A lei de controle deve minimizar o índice de desempenho, equação (2.85), onde as especificações são expressas através de formas quadráticas para o estado e o controle:

$$J_{RLQ} = x'(T)Mx(T) + \int_o^T (x'Qx + u'Ru)dt \tag{2.86}$$

e onde T é o horizonte de otimização, R é uma matriz definida positiva que pondera o controle, Q é uma matriz semidefinida positiva que pondera o estado e M é uma matriz semidefinida positiva que pondera o estado final.

O índice de desempenho é otimizado tendo como restrição o sistema dinâmico linear:

$$\dot{x} = Ax + Bu \tag{2.87}$$

A solução do problema RLQ, para T infinito e $M = 0$, definido pelas equações (2.86) e (2.87), é dada por:

$$u^* = -R^{-1}B'\bar{P}x \tag{2.88}$$

onde $\bar{P}$ é a solução da equação algébrica de *Riccati*:

$$0 = A'P + PA + Q - PBR^{-1}B'P \tag{2.89}$$

O desenvolvimento desta solução, equação (2.88), está apresentado no Apêndice (A). Esta é a solução do chamado problema do regulador em regime permanente, que por questão de simplicidade ao longo do texto será chamada de solução do problema do regulador linear quadrático.

2.3.2 As Matrizes de Ponderação Q e R

A escolha das matrizes de ponderação da função de custo J, equação (2.86), do projeto RLQ tem papel de grande importância na determinação dos ganhos $R^{-1}B'\bar{P}$ da lei de controle, equação (2.88), que minimiza J de forma a satisfazer os requisitos de projeto. Motivados por esta problemática, pesquisas têm sido conduzidas nas últimas três décadas no sentido de desenvolver métodos para determinação destas matrizes, tanto para efetivar alocação de pólos, (Medanic *et al.*, 1988), (Kawasaki e Shimemura, 1983) e (Graupe, 1972), como para efetivar a alocação de autoestruturas, (Harvey e Stein, 1978), (Stein, 1979), (Ochi e Kanai, 1996), (Choi e Seo, 1999*b*) e (Choi e Seo, 1999*a*), de forma determinística.

A tarefa de projetar controladores via problema RLQ, de forma a satisfazer especificações de desempenho nos domínios do tempo ou da frequência, relaciona-se diretamente com a seleção das matrizes de ponderação M, Q e R.

Diversos critérios foram desenvolvidos para seleção destas matrizes. Uma regra bastante simples, que regula o amortecimento de variáveis no domínio do tempo, é a escolha de matrizes de ponderação diagonais; o amortecimento de uma dada variável está relacionado com a magnitude do elemento da matriz diagonal correspondente à variável em questão; elementos diagonais de grande magnitude correspondem a um maior amortecimento. A seleção destas matrizes de ponderação para sistemas monovariável e multivariável é discutida por *Anderson e et al.*, (Anderson e Moore, 1990), tanto do ponto de vista de manipulação das ponderações do estado, quanto do controle.

Pesquisas desenvolvidas sobre a seleção das matrizes de ponderação conduziram ao estabelecimento de algumas variações bem definidas do problema RLQ. Por exemplo, o problema do *controle barato*, equação (2.90), é aquele em que ρ tende para zero na função objetivo J. Outro exemplo, o problema do controle terminal, equação (2.91), consiste em penalizar o estado terminal fazendo que o fator ρ seja diferente de zero e que a matriz de ponderação Q seja nula. Outra variação, problema do grau de estabilidade, equação (2.92), consiste em restringir as partes reais do autovalores do sistema em malha fechada no semi-plano complexo esquerdo de um valor α.

A seguir, apresenta-se as funções de custo modificadas, equações (2.90-2.92), que definem os problemas de *controle barato*, controle terminal e grau de estabilidade, respectivamente:

Função de custo para o problema do *controle barato*:

$$J_{Cbarato} = x'(T)Mx(T) + \int_t^T (x'Qx + \rho u'Ru)dt \tag{2.90}$$

Função de custo para o problema do controle terminal:

$$J_{Cterminal} = x'(T)\rho Mx(T) + \int_t^T u'Rudt \tag{2.91}$$

Função de custo para o problema do grau de estabilidade:

$$J_{Cestabilidade} = x'(T)Mx(T) + \int_0^\infty e^{2\alpha t}(x'Qx + u'Ru)dt \tag{2.92}$$

O surgimento de computadores de alto desempenho conduziu a um rápido desenvolvimento e à aplicação quase que imediata das técnicas de computação evolutiva, (Bäck, 1996), tal qual algoritmos genéticos, (Holland, 1975) e (Goldberg, 1989). Então, o desenvolvimento destas técnicas aliadas a unidades de processamento de alta velocidade promove uma nova alternativa para escolha das matrizes de ponderação do problema *RLQ*, especificamente, para o problema de alocação de autoestruturas, (Davis e Clarke, 1995) e (Bottura e Fonseca Neto, 1999*a*).

2.4 Otimização Multiobjetivo e Sistemas de Controle

A introdução do conceito de otimalidade de *Pareto* na solução de problemas de controle, por (Zadeh, 1963), abriu espaço para o desenvolvimento de pesquisa e aplicações da teoria de otimização multiobjetivo no projeto de sistemas de controle. *Dorato*, (Dorato, 1991), justifica a aceitação destas teorias devido ao fato de que certos problemas de projeto em engenharia devem satisfazer uma série de requisitos de desempenho que são conflitantes entre si. Além de contribuições relevantes, tais como: (Chang, 1966), (Chyung, 1967) e (Athans e Geering, 1973), para a teoria de controle multiobjetivo, cita-se nesta tese algumas referências que utilizam buscas aleatórias polarizadas, tais como: computação evolutiva e *simulated annealing*, e outros métodos como LMI's, (Whidborne *et al.*, 1997). A maior parte das contribuições em controle multiobjetivo relativas à década de 60 podem ser encontradas em (Salukvadze, 1979).

O problema de otimização multiobjetivo pode ser definido como um problema em que metas conflitantes devem ser atingidas (maximizar/minimizar) simultaneamente e condicionadas a certas restrições. Então, o problema de otimização multiobjetivo é formalizado como:

$$\begin{array}{c} \max f_1(x) \\ \vdots \\ \max f_k(x) \\ \min f_{k+1}(x) \\ \min f_n(x) \end{array} \tag{2.93}$$

$$s.a \qquad x \in S$$

onde $f_1(x) \ldots f_k(x)$ são os k objetivos que devem ser maximizados e $f_k(x) \ldots f_n(x)$ são os $n-k$ objetivos que devem ser minimizados na região S do espaço de busca.

A teoria e a metodologia desenvolvida por (Chankong e Haimes, 1983) é a que melhor se adequa para definir a estrutura de otimização multiobjetivo desenvolvida nesta pesquisa, tendo em vistas a alocação de autoestruturas. Estes definem o problema de otimização como um problema de decisão multiobjetivo, composto dos seguintes elementos: unidade geradora de decisão, situação de decisão, conjunto de objetivos, conjunto de atributos e regras de decisão.

Receber informação, gerar informação, processar a informação e produzir decisões são as funções da unidade geradora de decisão, que pode realizá-las com ou sem qualquer interferência do projetista. A definição de uma decisão, considerando eventos passados e atuais, tem por base um conjunto de regras que permite pontuar as várias alternativas de busca e a alternativa que atingir a maior pontuação é a eleita para guiar a busca.

2.5 Um Método de Desigualdades para AAE

O método das desigualdades desenvolvido por (Zakian e Al-Naib, 1973) para o projeto de sistemas dinâmicos e de controle consiste de uma formulação multiobjetivo para solucionar problemas de engenharia. Nesta formulação,

os índices de desempenho e restrições podem ser expressos em termos de desigualdades que representam diretamente uma dada situação.

Nesta tese, o problema de alocação de autoestruturas é formulado tendo por base o método das desigualdades e projetos RLQ, cujas variáveis independentes são as matrizes de ponderação Q e R. Esta formulação é uma generalização do método apresentado na seção (2.6).

Considere o sistema dinâmico e a sua representação em variáveis de estado, equação (2.76). O problema consiste em determinar pares de matrizes Q e R para o projeto RLQ de forma a satisfazer o conjunto de k desigualdades algébricas (2.94), que representam especificações e restrições de projeto:

$$f_i(Q,R) \leq C_i, \qquad i = 1,\ldots,k \tag{2.94}$$

onde C_i é um número real representando especificações e restrições de projeto e f_i é uma função real de Q e R.

Cada desigualdade $f_i(Q,R) \leq C_i$ define conjuntos (2.95) de pares de matrizes reais de dimensão $Q^{n\times n}$ e $R^{m\times m}$ relacionados com as matrizes Q e R, respectivamente.

$$S_{QR_i} = \{Q \geq 0 \ e \ R > 0 : f_i(Q,R) \leq C_i\} \qquad i = 1,\ldots,k \tag{2.95}$$

A interseção de todos os conjuntos S_{QR_i}, equação (2.96), significa que existem pares de matrizes Q e R que satisfazem simultaneamente as desigualdades dadas por (2.94).

$$S_{QR} = \cap_{i=1}^{k} S_{QR_i} \tag{2.96}$$

Comentários relevantes a respeito da natureza do método das desigualdades são discutidos em (Liu e Patton, 1998). O primeiro comentário afirma que o método fornece satisfabilidade e não desempenho ótimo como os métodos de otimização. O segundo comentário afirma que o método das desigualdades possui uma natureza interativa, fornecendo flexibilidade para o ajuste dos limites C_i das desigualdades (2.94) de forma a levar em consideração diversos conflitos entre os índices de desempenho $f_i(Q,R)$.

2.6 O Problema de AAE como um Problema RLQ

O problema é formulado considerando a determinação de ganhos do controlador por realimentação de estado que conduzirão a realização do objetivo principal do projeto: alocação da autoestrutura especificada via solução de problemas RLQ. Contudo, devido ao grande número de tentativas que devem ser feitas para determinar as matrizes de ponderação Q e R que satisfazem as especificações de projeto, e para usufruir de qualidades do projeto RLQ, tais como margens de estabilidade garantidas, decidiu-se utilizar algoritmos genéticos para realizar a busca de matrizes de ponderação que satisfaçam às especificações do projeto.

Considere um sistema dinâmico linear, equações (2.76-2.77), controlável:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{2.97}$$

O vetor de controle é dado por:

$$u = -Kx \tag{2.98}$$

Os ganhos do controlador K são dados pela solução da equação algébrica de *Riccati* (EAR), equação (2.100), para o problema de otimização linear quadrático, onde a lei de controle u é determinada, equação (2.98), através da minimização da função de custo de desempenho quadrático, equação (2.99),

$$J = \int_0^\infty [x'Qx + u'Ru]dt \tag{2.99}$$

sujeita à restrição (2.97). Para a equação algébrica de *Riccati*:

$$A'P + PA + Q - PBR^{-1}B'P = 0 \tag{2.100}$$

Q é a matriz de ponderação semidefinida positiva para os estados, R é a matriz ponderação definida positiva para os controles e P é a matriz que representa a solução única semidefinida positiva da EAR.

A solução $P = P(Q, R)$ fornece os ganhos do controlador $K = K(Q, R)$:

$$K = -R^{-1}B'P(Q, R) \tag{2.101}$$

Deste ponto de vista o problema de alocação de autoestrutura consiste na determinação da matriz de ganho $K(Q,R)$ que impõe o sistema de malha fechada especificado, equação (2.102), e que satisfaz às restrições de autoestrutura, relações (2.103) e (2.104):

$$\dot{x} = (A - BK(Q,R))x \tag{2.102}$$

As faixas do espectro dos autovalores de malha fechada devem satisfazer às especificações de projeto:

$$\lambda_{ei} \leq \lambda_{ci}(Q,R) \leq \lambda_{di}, \qquad i = 1,\ldots,n \tag{2.103}$$

onde λ_{ei} e λ_{di} são os limites dos *i-ésimos* autovalores à esquerda e à direita, respectivamente, para o *i-ésimo* autovalor λ_{ci} calculado e com seus *i-ésimos* autovetores à esquerda e à direita $L_i(Q,R)$ e $R_i(Q,R)$, respectivamente, satisfazendo:

$$S_i = S_i(Q,R) = \frac{\|L_i(Q,R)\|_2 \|R_i(Q,R)\|_2}{< L_i(Q,R), R_i(Q,R) >}, \qquad i = 1,\ldots,n \tag{2.104}$$

onde S_i é a sensibilidade do *i-ésimo* autovalor, (Wilkinson, 1965), $\|L_i(Q,R)\|_2$ e $\|R_i(Q,R)\|_2$ são as normas-2 dos autovalores à esquerda e à direita, respectivamente, e $< L_i(Q,R), R_i(Q,R) >$ é o produto interno dos autovetores.

Unindo a solução do problema RLQ, equação (2.101), e as restrições da autoestrutura, (2.103) e (2.104), têm-se como resultado o problema AAE formulado como um problema de otimização, que permite a determinação de um controlador $K(Q,R)$ através da aplicação de técnicas de busca aleatória. Desta forma, apresenta-se uma nova alternativa para alocar simultaneamente autovalores e autovetores, porque a formulação clássica do projeto RLQ não é adequada para solucionar este problema.

Uma nova função de custo pode ser a soma das sensibilidades dos autovalores, equação (2.105), onde a verificação da satisfabilidade da autoestrutura calculada é feita através das restrições, (2.103) e (2.104); desta maneira, a alocação de autoestruturas é possível via o controlador K obtido através da equação (2.101). Com esta nova função de custo, tem-se um problema de otimização multiobjetivo:

$$\min_{Q,R} \sum_{i=1}^{n} s_i(Q,R) \tag{2.105}$$

s.a

$$s_i(Q,R) \leq 1, \quad i = 1,...,n \tag{2.106}$$

$$\lambda_{ei} \leq \lambda_{ci}(Q,R) \leq \lambda_{di}, \quad i = 1,...,n \tag{2.107}$$

onde $s_i = S_i/\epsilon_i$ é a *i-ésima* sensibilidade normalizada em relação à *i-ésima* especificação de projeto: $\epsilon_i > 0$.

Formulações alternativas podem ser vistas em (Liu e Patton, 1998) e (Bottura e Fonseca Neto, 1999*b*). A equação (2.105) é utilizada como um índice de desempenho porque consegue expressar a evolução do processo de alocação como um todo; quando menor ou igual a n significa que a busca conseguiu atingir a satisfabilidade.

Capítulo 3

Otimizador Genético Paralelo

3.1 Introdução

As pesquisas desenvolvidas por *John Holland* e pesquisadores associados da Universidade de Michigan, Estados Unidos, tendo como ponto central dois objetivos, deram origem aos fundamentos dos algoritmos genéticos. Segundo (Goldberg, 1989) os objetivos foram: primeiro, a abstração e uma explicação rigorosa do processo adaptativo dos sistemas naturais; segundo, o projeto de algoritmos para sistemas artificiais que imitam importantes mecanismos dos sistemas naturais. Resultados desta pesquisa foram publicados em *Adaptation in Natural and Artificial Systems*, (Holland, 1975), que é considerado um dos trabalhos pioneiros na área de algoritmos evolucionistas junto com os trabalhos desenvolvidos por *Rechenberg*, em Berlin no anos 60, dando um enfoque em um tipo de algoritmo chamado estratégia evolutiva, (Bäck, 1996).

Os algoritmos genéticos estão classificados como um método de busca aleatória polarizado que tem por base os mecanismos da seleção natural, (Darwin, 1859), e da genética natural. Os autores, *Alba* e *Cota*, (Alba e Cotta, 1997), classificam os algoritmos genéticos (AG) como técnicas de busca aleatória guiada que fazem parte de um conjunto de métodos, tais como: busca tabu, *simulated annealing*, redes neurais e algoritmos evolutivos; estes pertencem ao conjunto dos algoritmos evolutivos, que por sua vez formam um conjunto de métodos tendo por base o conceito de evolução natural e são constituídos pelos seguintes métodos: programação genética, estratégias evolutivas, programação evolutiva e algoritmos genéticos; os algoritmos genéticos, por sua vez, são classificados como sequenciais e paralelos. Outros autores,

(Adenso *et al.*, 1996), classificam os *AG*, tendo por base (Barr *et al.*, 1995), como metaheurísticas sem memória que ainda podem ser classificados como algoritmos iterativos, heurísticos, de atuação isolada e critério de parada controlado pelo usuário. Estas classificações são duas diferentes visões de contextualização dos métodos; a primeira utiliza uma classificação generalizada dos princípios básicos de fundamentação das metodologias e a segunda utiliza uma analogia fragmentada, no sentido de comparar como estes tratam eventos que são comuns entre si, para classificar os métodos de busca.

Os conceitos básicos e definições de termos da genética artificial, análogos aos de sistemas naturais, são descritos na seção (3.2). A analogia é feita através das matrizes de ponderação Q e R modeladas como indivíduos cromossômicos, explorando os conceitos de cromossomos, genótipos, fenótipos, genes e outros termos que foram criados para sintetizar idéias que fazem parte do algoritmo proposto.

Os algoritmos genéticos são fortes candidatos a serem paralelizados porque possuem características intrínsecas a sua formulação que conduzem a uma fácil paralelização; estas características são acentuadas em nível de operações genéticas e cálculo da função de *fitness*. A seção (3.3) salienta por que estes algoritmos possuem um paralelismo natural e apresenta uma classificação dos *AG* paralelos, que fornece os elementos básicos para classificar o modelo de algoritmo proposto nesta tese.

A seção (3.4) apresenta os fundamentos do algoritmo genético proposto. Os seus conceitos básicos são discutidos, tendo como referência um conjunto constituído por três elementos: seleção, conjunto de operações cromossômicas e procedimento de adequabilidade ou *fitness*. O elemento *seleção* envolve as escolhas do tipo de operação cromossômica, dos indivíduos que serão operados em nível de cromossomos e da quantidade de indivíduos que podem ser incluídos na população permanente; as escolhas têm por base um gerador de números pseudo-aleatórios. O elemento *operações cromossômicas* é constituído pelas operações duplicação, *crossover*, mutação e visitante; estas operações são modeladas em nível dos elementos das matrizes de ponderação do problema *RLQ*. O elemento procedimento de *fitness* envolve três fases: pontuação de indivíduos, ordenação destes indivíduos de acordo com o valor atribuído pela pontuação e inclusão deles na população permanente de acordo com um critério estabelecido pela quantidade máxima de indivíduos que podem ser inseridos nesta população.

O modelo do algoritmo genético paralelo proposto é caracterizado por buscas independentes, granularidade grossa e hierárquico. A seção (3.5) abor-

da as principais características deste algoritmo e apresenta sua classificação de acordo com as caracterizações apresentadas na seção (3.3).

3.2 Conceitos Básicos e Definições

A analogia entre a genética natural e a artificial apresentada por (Goldberg, 1989), é comentada nesta seção para esclarecer alguns dos termos da computação evolutiva dentro do contexto da modelagem genética das matrizes de ponderação Q e R do problema RLQ. Estas matrizes são modeladas como indivíduos-QR que fazem parte de uma população de matrizes. Apresenta-se também a definição de alguns termos criados no decorrer do desenvolvimento desta tese.

A Figura (3.1) representa um modelo para matrizes de ordem 2. As matrizes Q e R são transformadas em *strings*, cadeias de caracteres, dos tipos Q e R, respectivamente. Estes *strings* correspondem aos cromossomos e os elementos das matrizes correspondem ao genes da genética natural. A quantidade de genes de um cromossomo é $(n^2 + n)/2$, porque as matrizes Q e R são simétricas. Os elementos das matrizes podem ser representados nas bases numéricas, decimal, hexadecimal, binária, etc; este tipo de representação é genericamente chamado de alfabeto do modelo cromossômico.

A união de genes forma os cromossomos; cada gene possui um valor que é chamado de alelo e uma posição ou *locus*. Na genética artificial, para o caso específico das matrizes, o valor de cada elemento representa um alelo e seus índices (i, j) representam a posição. Logo, o conjunto constituído pelos elementos alelo e *locus* caracteriza um gene artificial que representa um elemento das matrizes de ponderação.

Um cromossomo do tipo Q unido a um do tipo R forma a prescrição genética total para a construção e operação de algum organismo. Esta combinação é chamada de genótipo, que equivale a uma estrutura na genética artificial. O meio ambiente é caracterizado por funções que são chamadas de funções de *fitness* ou de adequabilidade; nos sistemas artificiais os indivíduos formados a partir da interação com o meio ambiente são chamados de fenótipos. No contexto desta tese, os termos função de *fitness* e algoritmo de *fitness* possuem o mesmo significado.

Uma população de indivíduos-QR é constituída por um conjunto de indivíduos-QR ou indivíduos cromossômicos que são formados através da união de cromossomos dos tipos Q e R, Figura (3.2). A Figura (3.2) representa

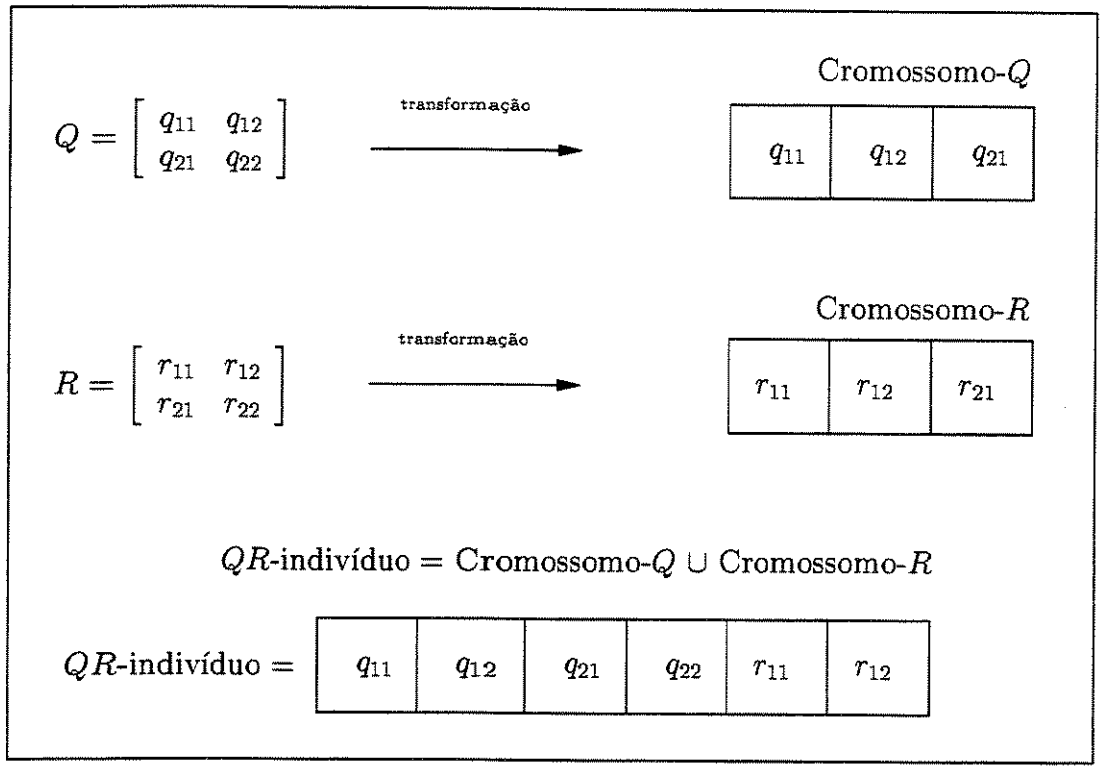

Figura 3.1: Modelo genético artificial de ordem 2 para as matrizes Q e R

uma população-QR, ou população de matrizes de ponderação, de ordem 2 com n indivíduos.

O termo *ciclo de busca* significa uma série de passos sequenciais que envolve: geração das matrizes de ponderação, cálculos intermediários, avaliação da função de *fitness* e verificação de critérios de parada. A Figura (3.4), subseção (3.4.2), representa os passos do *ciclo de busca* e a quantidade de repetições do ciclo caracteriza o seu tamanho.

Durante o processo de busca, cada indivíduo é pontuado com valores escalares determinados na fase de pré-*fitness* ou de cálculos intermediários; os passos do pré-*fitness* são: determinação dos ganhos do controlador via solução da equação de *Riccati*, (Laub, 1979), montagem do sistema de malha-fechada e computação dos autovalores e dos autovetores deste sistema. A função de *fitness* verifica restrições e depois ordena os indivíduos de acordo com o grau

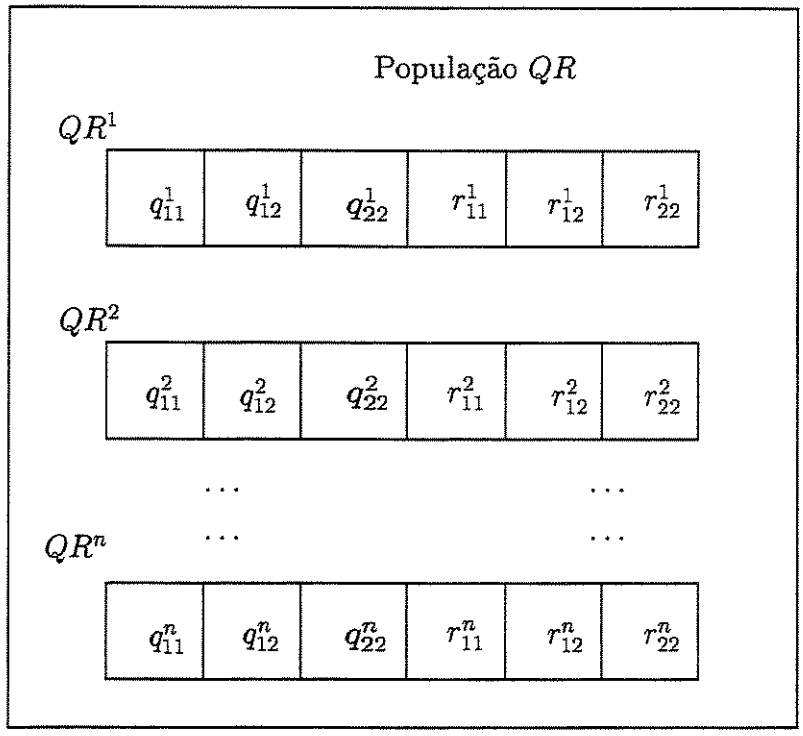

Figura 3.2: População de indivíduos QR de ordem 2.

de satisfabilidade das autoestruturas computadas em relação à autoestrutura especificada; isto significa que esta ordenação estipula o grau de adequabilidade; este grau traduz a chance de sobrevivência de cada indivíduo no meio ambiente, onde a quantidade de indivíduos que são escolhidos para participar de uma população é limitada, pois os indivíduos-QR competem entre si para sobreviver durante as iterações do *ciclo de busca.*

Dois tipos de população fazem parte do processo de busca. Uma é chamada de população permanente, porque sempre é constituída de um número fixo de indivíduos e é montada no final da fase de ordenação. O segundo tipo, chamado de população transitória, é construído com todos os indivíduos da população permanente do passo anterior e com todos os novos indivíduos formados a partir das operações genéticas; esta população é sempre maior que a permanente.

O termo *clonar* significa que uma replica do melhor indivíduo da popu-

lação permanente é feita através da operação genética de duplicação.

Um *schema* (plural: *schemata*) refere-se as cadeias de caracteres que tem valores idênticos em certos *loci*. A forma de um *schema*: consiste de um padrão em que os *genes* comuns são indicados explicitamente e o símbolo "*", ou outro qualquer, indica a parte irrelevante da cadeia de caracteres. Por exemplo, $1*101$ é um padrão para as cadeias de caracteres $\{10101, 111101\}$. De forma geral, os *schemata* são padrões que especificam, parcialmente, uma solução do espaço genotípico.

O termo *mating* é utilizado no contexto das operações cromossômicas, exclusivamente para a operação *crossover*; significa que dois indivíduos-QR foram escolhidos para serem combinados e como resultado tem-se dois novos indivíduos, ou *off-springs*. A efetivação da combinação, ou *mating*, ocorre no *mating pool*.

3.3 AG-paralelos

Os algoritmos genéticos (AG) sequenciais têm sido aplicados (Kristinsson e Dumont, 1992), (Marra e Walcott, 1996), (Michalewicz *et al.*, 1992), (Porter e Passino, 1994), (Morales, 1995), (Dangprasert e Avatchakorn, 1996) com sucesso para resolver problemas de controle. Contudo, o tempo gasto na avaliação da função de *fitness*, necessidade do uso de população de grande tamanho e convergência prematura são algumas desvantagens que motivam o desenvolvimento de algoritmos genéticos paralelos (AGP). As necessidades de explorar o paralelismo natural dos AG e de criar modelos altamente eficientes também conduzem à paralelização destes algoritmos.

O alto grau de paralelismo existente nos algoritmos genéticos é inerente à sua estrutura de *problem solver*. Isto pode ser exemplificado sob dois aspectos: o primeiro refere-se às operações genéticas, porque feita a seleção do conjunto de indivíduos que serão acasalados, criados, duplicados e transformados através das operações genéticas, estas podem ser feitas simultaneamente; o segundo refere-se à avaliação da função de *fitness* que também pode ser feita simultaneamente logo após a geração de qualquer novo indivíduo. Estes eventos podem ocorrer de forma paralela, síncrona ou assíncrona.

Historicamente, o alto grau de paralelismo dos algoritmos genéticos foi observado por (Holland, 1962). Com exceção da pesquisa desenvolvida por (Grefenstette, 1981), em que classifica os AG paralelos como: mestre-escravo, distribuído e rede, pouca importância havia sido dada à paralelização destes

algoritmos, (Goldberg, 1989). Atualmente, *ad hoc* 1999, existe um crescente direcionamento dos pesquisadores voltado para o desenvolvimento de teorias e de aplicações dos AG paralelos, que pode ser comprovado pelos resultados apresentados em (Cantú-Paz, 1999), (Huntley e Brown, 1996), (Pettey e Leuze, 1989), (Mühlenbein, 1989), (Brown *et al.*, 1989), (Gorges-Schleuter, 1989), (Manderick e Spiessens, 1989) e (Tanese, 1989).

O processamento paralelo tem como paradigma básico o lema dividir-para-conquistar, ou seja, a solução de um problema em questão é dividida em tarefas que podem ser executadas independentemente e seus resultados são unidos para fornecer uma solução. Quanto ao processamento paralelo de algoritmos genéticos, os autores: (Cantú-Paz, 1998), (Koza, 1992) e (Alba e Cotta, 1997), apresentam várias taxonomias, entretanto todas elas têm por base a distribuição: da população e/ou da função de *fitness* e/ou de operadores genéticos. A seguir, comenta-se algumas das classificações propostas para os algoritmos genéticos.

Algoritmos genéticos: mestre-escravo global com população única, granularidade fina com população única e granularidade grossa com população múltipla são os três principais tipos de modelos de paralelização dos AG's caracterizados por (Cantú-Paz, 1998). No primeiro tipo, a paralelização ocorre em nível da função de *fitness* e o termo *global* refere-se ao envolvimento de toda a população no processo de seleção e de operações genéticas. No segundo, os *AG's* granularidade fina são caracterizados por terem uma população espacialmente estruturada e serem adequados para implementações em computadores massivamente paralelos. No terceiro tipo, granularidade grossa, são caracterizados por múltiplas populações e pela migração de indivíduos; devido a certas características de sua implementação são conhecidos por: *AG's* de paralelização *island* porque as subpopulações estão relativamente isoladas, *AG's* distribuídos porque normalmente são implementados em máquinas *MIMD* e *AG's* de granularidade grossa porque a relação computação/comunicação é relativamente alta. Uma combinação dos três principais tipos de paralelização produz os chamados *AG's* paralelos hierárquicos que são caracterizados por possuirem múltiplas populações no alto nível e com os *AG's* paralelos com populações únicas no nível inferior.

Uma classificação baseada no modelo, na distribuição da população e na implementação é apresentada em (Alba e Cotta, 1997). No modelo paralelo são todos globais e são distintos em relação à migração, que podem ser com ou sem migração ou indivíduos podem ser adicionados na população permanente. Em relação à migração, esta pode ser de granularidade fina ou

grossa. A distribuição da população pode ser centralizada ou distribuída. As implementações podem ser: a) *farming*, onde um processador central realiza o processo de seleção e os processadores escravos realizam as operações genéticas e a avaliação da função de *fitness*; b) celulares, conhecida como AG massivamente paralelos; nesta implementação, os indivíduos de uma única população são distribuídos segundo uma disposição espacial de forma tal que as operações genéticas sejam efetuadas em paralelo e cada indivíduo interaja somente na vizinhança adjacente.

Apesar da classificação apresentada por (Koza, 1992) referir-se à programação genética, quando vista em termos gerais, no escopo de computação evolutiva, a idéia básica pode ser extrapolada para os algoritmos genéticos. Então, a citada referência classifica a paralelização em duas abordagens básicas. A primeira abordagem, segue os mesmos princípios do terceiro tipo de paralelização caracterizada por (Cantú-Paz, 1998); a população é dividida em *demes* (subpopulações), e cada *deme* está associada a um processador e separadamente o AG opera em cada *deme*; durante o processo de busca ocorre uma troca de indivíduos entre as subpopulações. Na segunda abordagem, existe uma única população para cada processador e não existe troca de indivíduos entre populações. Em certos problemas, a complexidade do cálculo da função de *fitness* pode demandar um tempo de processamento proibitivo para o AG fornecer uma solução. Para contornar este problema, *Koza*, (Koza, 1992), sugere três métodos para paralelizar esta função que são referenciados como níveis de paralelização: de casos de *fitness*, de indivíduos e de processamentos independentes.

3.4 Modelo proposto para o AG

Esta seção apresenta os elementos básicos que compõem o algoritmo genético paralelo desenvolvido para a alocação de autoestruturas de sistemas dinâmicos através do projeto RLQ. Especificamente, tem-se por objetivo a determinação das matrizes de ponderação Q e R deste método; esta especificidade faz-se notada pelo modelamento destas matrizes, onde cada par de matrizes Q e R constitui um indivíduo-cromossômico no modelo genético artificial. O algoritmo proposto, de uma forma geral, procura guiar a busca definindo o comportamento do processo; isto significa que o AG proposto é dedicado à especificidade do problema não só em nível de modelamento, mas também através da formulação de mecanismos de decisão projetados para guiar a bus-

ca das matrizes de ponderação; este ponto constitui a diferença essencial entre o algoritmo proposto e o algoritmo genético canônico, (Altenberg, 1995), que basicamente faz uma busca polarizada do espaço de solução, enquanto que o AG aqui proposto possui a característica adicional de definir direções de busca dentro do espaço de solução. Os mecanismos que direcionam a busca e os seus fundamentos são descritos no capítulo (4), porque se inserem dentro do contexto de estratégias de direcionamento de buscas.

3.4.1 Modelagem das Matrizes Q e R

As matrizes Q e R são modeladas como cromossomos. Duas bases numéricas são utilizadas para representar os seus alelos; uma binária e a outra decimal. Cada matriz é representada por um cromossomo, equações (3.1-3.2), cada par de matrizes Q e R é chamado de indivíduo-QR, equação (3.3), e cada conjunto de pares de matrizes constitui uma população, equações (3.4-3.6). As relações (3.1-3.6) formalizam de forma genérica o modelo proposto para as matrizes de ponderação Q e R de ordem n:

$$Q_{cromossomo} = q_{11} \cup q_{12} \ldots \cup q_{1n} \ldots \cup q_{22} \cup q_{2n} \ldots \cup q_{nn} \tag{3.1}$$

$$R_{cromossomo} = r_{11} \cup r_{12} \ldots \cup r_{1n} \ldots \cup r_{22} \cup r_{2n} \ldots \cup r_{nn} \tag{3.2}$$

onde $q_{i,j}$ e $r_{i,j}$, $i = 1, \ldots, (n^2+n)/2$ e $j = 1, \ldots, (n^2+n)/2$, representam os elementos das matrizes Q e R, respectivamente, que são chamados de alelos e possuem significado análogo ao da genética natural. Os elementos Q e R são representados em bases numéricas binárias ou decimais.

$$QR_{cromossomo} = q_{11} \cup q_{12} \ldots \cup q_{nn} \cup r_{11} \cup r_{12} \ldots \cup r_{nn} \tag{3.3}$$

$$\begin{aligned}
QR^1_{cromossomo} &= q^1_{11} \ldots \cup q^1_{nn} \cup r^1_{11} \ldots \cup r^1_{nn} && (3.4)\\
QR^2_{cromossomo} &= q^2_{11} \ldots \cup q^2_{nn} \cup r^2_{11} \ldots \cup r^2_{nn} && (3.5)\\
\ldots &= \ldots \quad \ldots \quad \ldots &&\\
QR^k_{cromossomo} &= q^k_{11} \ldots \cup q^k_{nn} \cup r^k_{11} \ldots \cup r^k_{nn} && (3.6)
\end{aligned}$$

Os elementos das matrizes Q e R são associados com os genes da genética natural. Para o caso de uma representação binária, a magnitude do alelo depende da quantidade de *bits*, que determina os limites numéricos máximos

e mínimos de cada elemento. A Figura (3.3) mostra como os elementos q_{ij} ou r_{ij} podem ser modelados na genética artificial em uma base binária, supondo que seus limites estão na seguinte faixa $0 \leq (q_{ij}\ e\ r_{ij}) \leq 255$. Para este caso, tem-se oito *bits* reservados para a parte inteira e oito *bits* reservados para a parte decimal. Os bits mais à esquerda representam a parte inteira e os mais à direita representam a parte fracionária. O número total de *bits* necessário para representar uma matriz, Figura (3.3), é dado pelo número de *bits* utilizados para representar numericamente um elemento da matriz, na precisão especificada, multiplicado pela quantidade, $(n^2+n)/2$, de elementos de cada matriz, considerando iguais os limites das magnitudes dos seus elementos.

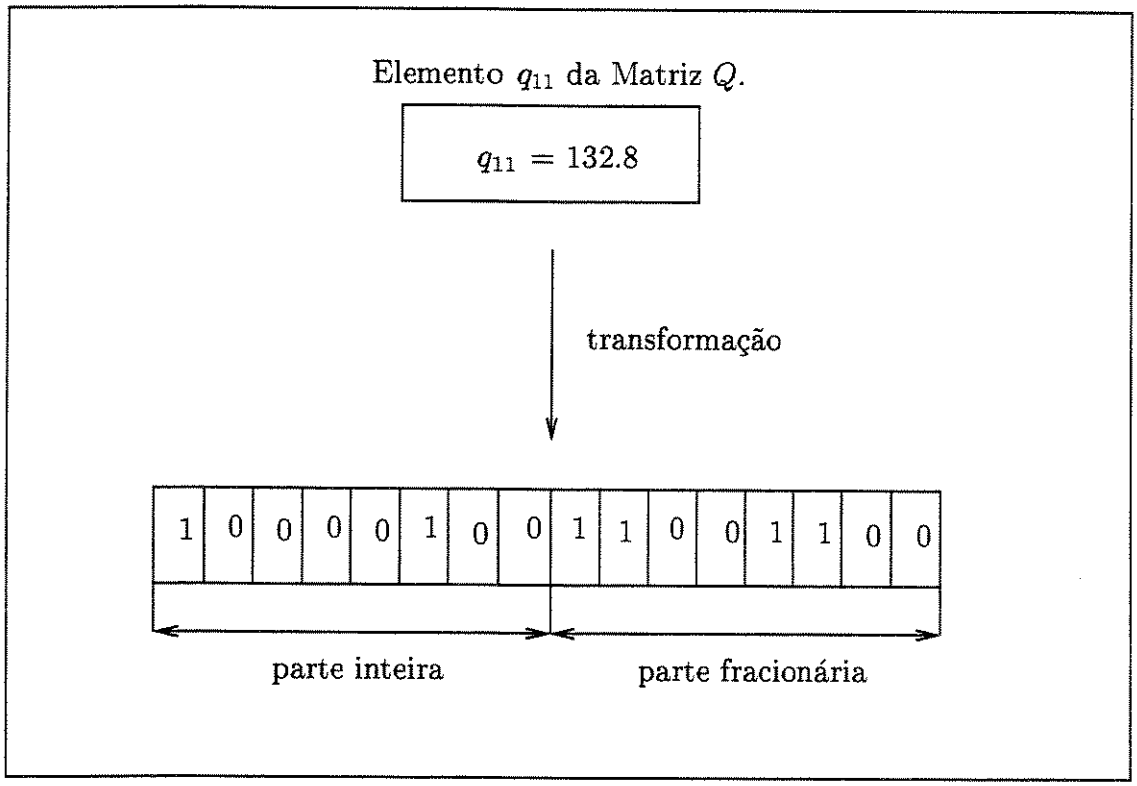

Figura 3.3: Modelagem do Elemento q_{11} da Matriz Q.

3.4.2 As Operações Genéticas

O procedimento para operações genéticas é basicamente constituído de cinco estágios sequenciais, que podem ser enumerados por ordem de ocorrência da seguinte maneira: seleção do tipo de operação cromossômica, seleção dos indivíduos que sofrerão ação das operações cromossômicas, efetivação das operações cromossômicas, avaliação dos novos indivíduos e seleção da quantidade de indivíduos para compor a população permanente. As operações genéticas cromossômicas são duplicação, *crossover* (recombinação ou cruzamento), mutação e visitante. A avaliação pode ser constituída de quatro fases: transformações de indivíduos-QR em matrizes decimais de ponderação (só para a implementação na base numérica binária), pré-*fitness*, verificação de *fitness* e escolha dos melhores indivíduos para compor a nova população. A seguir, descrevem-se os conceitos e os modelos dos cinco estágios sequenciais.

A operação de Duplicação

A operação de duplicação não precisa ser avaliada porque esta operação apenas verifica qual é o indivíduo que possui o pior valor de *fitness*. Este é removido e substituído por um *clone* do indivíduo mais forte da população. Então,

$$q_{ij}^{novo} = q_{ij}^{melhor} \qquad (3.7)$$
$$i = 1, \ldots, (n^2+n)/2; \quad j = 1, \ldots, (n^2+n)/2$$

$$r_{ij}^{novo} = r_{ij}^{melhor} \qquad (3.8)$$
$$i = 1, \ldots, (n^2+n)/2; \quad j = 1, \ldots, (n^2+n)/2$$

A operação de Crossover

O procedimento de operações de *crossover* implementa tipos de combinações entre dois indivíduos que dependem do alfabeto, base numérica, utilizado na representação dos genes. Para o caso de uma representação binária, o algoritmo tem opção de efetuar três tipos de recombinação, que são: *X-over* de ponto único, *X-over* de ponto duplo e *X-over* uniforme. Para o caso de uma representação decimal, o procedimento implementa uma combinação linear

dos indivíduos, em que cada alelo resultante é constituído da combinação de dois alelos de indivíduos escolhidos por um dos métodos de seleção utilizados durante o *ciclo de busca* das matrizes de ponderação Q e R.

As equações (3.9-3.10) representam a operação de *crossover* entre alelos, na base decimal, para gerar novos cromossomos dos tipos Q e R, respectivamente, cromossomos estes montados a partir dos cromossomos Q e R de um indivíduo l, QR^l, e dos cromossomos Q e R de um indivíduo k, QR^k.

A geração de um cromossomo Q a partir da combinação de dois indivíduos l e k, respectivamente, para matrizes de ordem n é dada por:

$$\begin{aligned} q_{ij}^{novo} &= q(t)_{idad} q_{ij}^{l} - |1 - q(t)_{idad}| q_{ij}^{k} \\ & \qquad i = 1, \ldots, (n^2+n)/2; \quad j = 1, \ldots, (n^2+n)/2 \end{aligned} \tag{3.9}$$

onde $q(t)_{idad}$, $t = 1, \ldots, nq_{idad}$, é um parâmetro que pondera os valores de alelos dos cromossomos do tipo Q e que pode variar de acordo com a idade da população. A variação deste parâmetro é amplamente explorada no sentido de contribuir no direcionamento da busca das matrizes de ponderação.

As consequências do ajuste de parâmetros em algoritmos de computação evolutiva são investigadas por (Eiben *et al.*, 1999), que também propõe uma taxonomia em relação ao tipo de mecanismos de ajuste dos parâmetros; estes enfatizam que o ajuste de certos parâmetros provoca substanciais melhorias no desempenho de algoritmos baseados em computação evolutiva; estas observações foram também notadas, a *priori*, por (Bottura e Fonseca Neto, 1999*b*) e exploradas por (Bottura e Fonseca Neto, 2000). Os elementos q_{ij} da matriz Q são análogos aos alelos da genética natural e cada um deles assume um valor numérico. A quantidade de variações que o parâmetro $q(t)_{idad}$ pode sofrer é limitada pela quantidade de buscas que o algoritmo realiza para determinar as matrizes Q e R; então, $nq_{idad} \leq n^{\underline{o}}$ de buscas.

De forma similar, as mesmas considerações são válidas para um cromossomo do tipo R. Então, o modelo da operação *crossover* decimal para cromossomos-R é dada por:

$$\begin{aligned} r_{ij}^{novo} &= r(t)_{idad} r_{ij}^{l} - |1 - r(t)_{idad}| r_{ij}^{k} \\ & \qquad i = 1, \ldots, (n^2+n)/2; \quad j = 1, \ldots, (n^2+n)/2 \end{aligned} \tag{3.10}$$

onde o parâmetro $r(t)_{idad}$, $t = 1, \ldots, nr_{idad}$, atua em cromossomos do tipo R e pode variar de acordo com a idade da população. Os elementos r_{ij} representam os valores dos alelos do cromossomo R.

A combinação de m indivíduos tem como resultado m indivíduos-QR. De uma forma geral, os m indivíduos-QR gerados de operações de *crossover* decimal são representados como:

$$QR^{novo^{lk}} = Q^{novo^{lk}} \cup R^{novo^{lk}}, \quad lk = 1, \ldots, m \tag{3.11}$$

A Operação de Mutação

Além do procedimento de mutação binária, dois procedimentos de mutação decimal foram desenvolvidos; um deles implementa a mutação global e o outro implementa a mutação local; eles foram inspirados por (Marrison e Stengel, 1997).

O procedimento de mutação binária atua de forma a trocar os valores dos *bits* de alelos que são escolhidos aleatoriamente.

Os procedimentos para operação de mutação decimal são implementados da seguinte maneira: a mutação local consiste de uma modificação em todos os alelos de indivíduos escolhidos aleatoriamente de acordo com o princípio de seleção natural de *Darwin*. A equação (3.12) representa esta mutação para um cromossomo Q de um indivíduo l e o valor de cada alelo é modificado de maneira multiplicativa:

$$q_{ij}^{novo} = q_{ij}^{l} b^{x_{local}} \tag{3.12}$$
$$i = 1, \ldots, (n^2+n)/2 \quad e \quad j = 1, \ldots, (n^2+n)/2$$

onde q_{ij}^{novo} representa o valor do alelo do cromossomo do novo indivíduo Q. O elemento q_{ij}^{l} representa o valor do alelo do cromossomo-Q do indivíduo l. O valor b é a base determinística do multiplicador exponencial, $b > 1$, e x_{local} é o expoente aleatório, $0 \leq x_{local} \leq 1$.

As mutações locais para cromossomos do tipo R ocorrem de maneira similar à mutação dos cromossomos do tipo Q. Então, após m-mutações locais para um ciclo de busca têm-se m novos indivíduos-QR, equação (3.13).

$$QR^{novo^{l}} = Q^{novo^{l}} \cup R^{novo^{l}}, \quad l = 1, \ldots, m \tag{3.13}$$

O procedimento para mutação decimal global consiste da escolha de indivíduos aleatoriamente, tal qual na mutação local, e para estes indivíduos

os valores de cada alelo são incrementados de um valor que é o produto de um número aleatório pela diferença entre os alelos correspondentes do pior e do melhor indivíduo da população atual; a equação (3.14) representa esta mutação para um cromossomo do tipo Q de um indivíduo l:

$$q_{ij}^{novo} = q_{ij}^{l} + x_{global}|q_{ij}^{melhor} - q_{ij}^{pior}| \qquad (3.14)$$
$$i = 1, \ldots, (n^2+n)/2 \quad e \quad j = 1, \ldots, (n^2+n)/2$$

onde os elementos q_{ij}^{novo}, q_{ij}^{l} e x_{global} são definidos tal qual na mutação local. Os elementos q_{ij}^{melhor} e q_{ij}^{pior} representam os alelos dos cromossomos Q dos indivíduos que apresentam o melhor e o pior valores dados pela função de *fitness*, respectivamente, da população permanente atual. Um modelo similar é utilizado para cromossomos-R. Então, após m mutações globais dos cromossomos dos tipos Q e R, têm-se m indivíduos resultantes desta operação e a sua representação é similar à equação (3.13).

A operação Visitante

O operador *visitante* insere novos indivíduos que não possuem qualquer relação com a população atual e a probabilidade de ocorrência desta operação é muito pequena. Estes indivíduos são gerados de forma aleatória. Este operador tem por objetivo retirar a população da saturação e aumentar a diversidade do material genético artificial, isto é, retirar a população de ótimos locais e permitir uma melhor exploração do espaço de busca.

A Função de Fitness

A função de *fitness* ou de adequabilidade tem por objetivo apontar os indivíduos pertencentes à população permanente atual, juntamente com os criados a partir das operações genéticas, que estão mais aptos a sobreviver ou a fazer parte da próxima população permanente. Em suma, diz-se que a função de *fitness* pontua e seleciona indivíduos que melhor se adequam ao meio ambiente. A função de *fitness* representa o meio ambiente para uma determinada espécie.

Dentro do contexto do desenvolvimento deste trabalho a função de *fitness* é constituída de duas fases. A primeira fase, pontuação e ordenação, consiste da avaliação dos indivíduos que foram gerados a partir das operações

genéticas; todo indivíduo possui uma característica que permite verificar o quanto ele é adequado para ser um membro permanente da população atual; esta avaliação é feita numericamente através de um dada função e este número é uma marca da qualidade do indivíduo indicando o grau de satisfabilidade. A segunda fase, formação da população permanente, consiste da seleção propriamente dita; os indivíduos são escolhidos para compor a população e são ordenados de acordo com o grau de satisfabilidade. Os que melhor satisfazem um determinado índice são selecionados para compor a população permanente atual.

Mostra-se o procedimento da função de *fitness*, Figura (3.4), como um módulo do otimizador genético, antes de apresentar em detalhes cada passo deste algoritmo. A Figura (3.4) representa o otimizador genético salientando os passos para o cálculo da função de *fitness* dentro de um contexto generalizado, isto é, a geração das matrizes de ponderação que são criadas através das operações cromossômicas, os cálculos intermediários, a função de *fitness* e os critérios de parada. A funcionalidade de cada bloco, pré-*fitness* e função de *fitness*, é descrita a seguir: o primeiro bloco monta os coeficientes da equação algébrica de *Riccati* (EAR) tendo como variável independente as matrizes de ponderação, o segundo bloco representa o cálculo da lei de controle via solução P da EAR, o terceiro bloco representa o sistema de malha fechada, o quarto bloco representa o cálculo da autoestrutura do sistema de malha fechada, o quinto bloco representa a fase de pontuação e de ordenação dos novos indivíduos-QR e o sexto bloco representa a formação da nova população permanente.

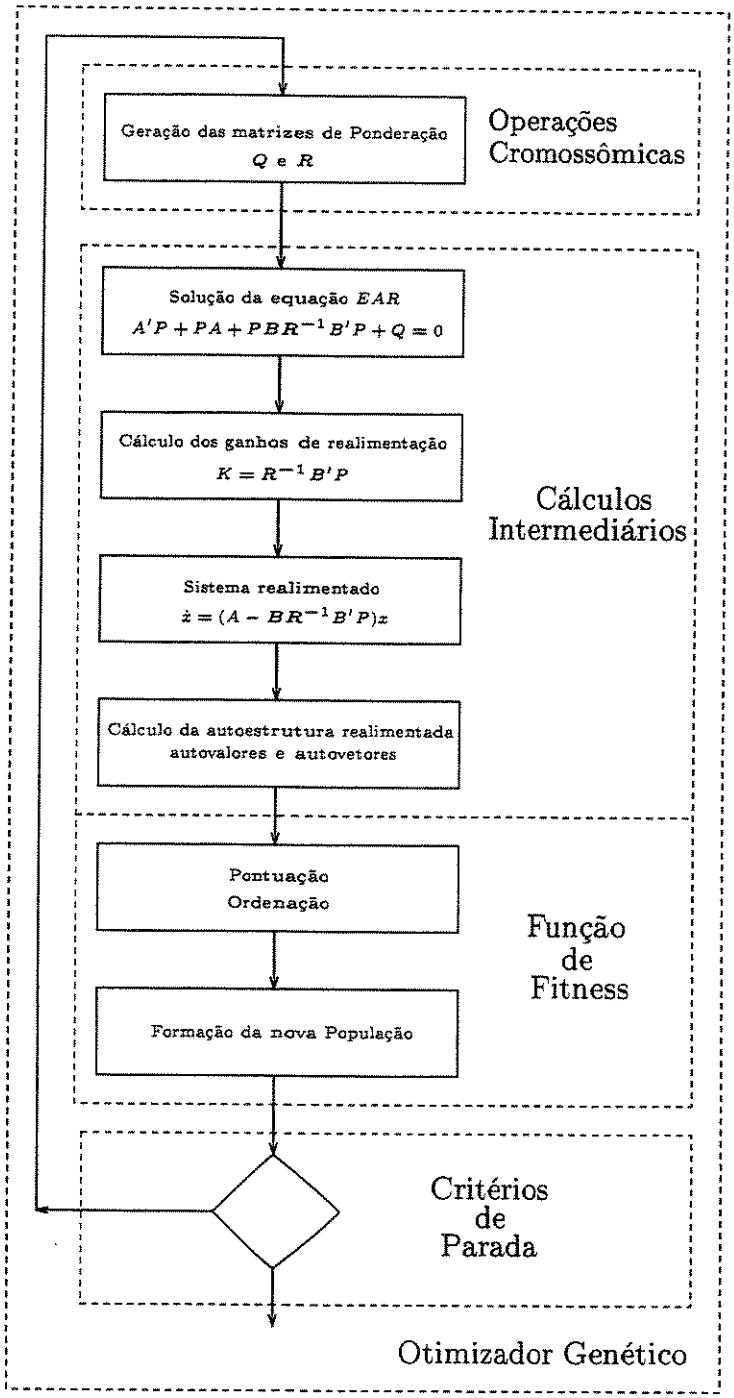

Figura 3.4: Passos para o Cálculo da função de **Fitness**.

A seguir, apresenta-se o procedimento de *fitness* tendo como ponto de partida a fase de pontuação dos indivíduos e como ponto final a inclusão de novos indivíduos na população permanente:

- Parâmetros que representam o melhor indivíduo da população inicial:
  - Menor função de desempenho entre todas as maiores funções de desempenho da população inicial, Δ_{atual};
  - Soma das funções de desempenho normalizadas, E_{atual};
  - Controlador K_{atual}.
- k-ésimo passo do procedimento de *fitness*:

1- Cálculo das funções de desempenho normalizadas:

$$s_{ij}(K_j^k) = \frac{S_{ij}(K_j^k)}{\epsilon_i}, \quad i = 1, 2, ..., n; j = 1, 2, ..., m \tag{3.15}$$

2- Máximo das funções de desempenho normalizadas:

$$\Delta_j^k = \max_i \{s_{ij}(K_j^k)\}, \quad i = 1, 2, ..., n; j = 1, .., m \tag{3.16}$$

3- Soma normalizada das funções de desempenho:

$$E_j^k = \sum_{i=1}^{n} s_{ij}(K_j^k), \quad i = 1, 2, ..., n; j = 1, .., m \tag{3.17}$$

4- Avaliação dos limites:

$$(a) \quad \Delta_j^k < \Delta_{atual}^{k-1}, \qquad j = 1, .., m \tag{3.18}$$

ou

$$(b) \quad \Delta_j^k = \Delta_{atual}^{k-1} \ e \ E_j^k < E_{atual}^{k-1}, \quad j = 1, .., m \tag{3.19}$$

5- Se uma das condições (3.18) ou (3.19) é satisfeita:

$$K_{atual}^k = K_{jo}^k \tag{3.20}$$

$$\Delta_{atual}^k = \Delta_{jo}^k \tag{3.21}$$

$$E_{atual}^k = E_{jo}^k \tag{3.22}$$

As fases de pontuação dos indivíduos de uma nova geração são descritas pelas equações (3.15-3.17), que representam o cálculo das sensibilidades normalizadas, a determinação da maior sensibilidade normalizada para cada indivíduo e o cálculo da função de custo; as relações (3.18-3.19) constituem a fase de seleção de novos indivíduos na população permanente; as equações (3.20-3.22) efetivam a inclusão de novos indivíduos na população permanente se e somente se as relações (3.18-3.19) são satisfeitas.

As equações e as relações citadas no parágrafo anterior compõem o procedimento de *fitness*, constituído pelas fases de pontuação, seleção e de inclusão de novos indivíduos-QR na população permanente, que tem por base a equação (2.105) e as relações (2.106-2.107) da formulação do problema, capítulo (2), secção (2.6). De uma maneira mais abrangente, o procedimento descrito tem por base um método para reduzir a sensibilidade dos autovalores em relação a variações dos autovetores, (Wilkinson, 1965), um método de desigualdades para AAE, seção (2.5), e o projeto RLQ. Modelos de funções de *fitness* mais sofisticados são apresentados no capítulo (4), seção (4.4), onde apresenta-se um conjunto de cinco funções que levam em consideração o critério da otimalidade de *Pareto*, articulação de preferências e outras estratégias para direcionar a busca.

A avaliação da população inicial é feita por razões óbvias; no caso visando a redução do esforço computacional, antes da sua distribuição. Esta avaliação inicial é caracterizada pela inicialização dos parâmetros que permitem definir o nível de qualidade do indivíduo (controlador) para o processo de seleção. Estes parâmetros, conforme definidos nas equações (3.16-3.17) do procedimento de *fitness*, são: a maior função de sensibilidade normalizada para o *j-ésimo* indivíduo-QR, Δ_j, a soma das funções de desempenho normalizadas, E_j, e os ganhos do controlador, K_j que é proveniente da solução da EAR. Estes três parâmetros caracterizam um indivíduo no sentido de que os dois primeiros fornecem informações sobre o processo de busca e o último fornecerá informações sobre o desempenho do controlador quando implementado no modelo do sistema dinâmico.

Métodos de Seleção

No contexto deste trabalho, os métodos de seleção são classificados de acordo com a finalidade do processo de escolha. Existem três processos de seleção que são nitidamente distintos durante a busca das matrizes de ponderação, e todos eles são dependentes de um gerador de números pseudo-aleatórios.

O primeiro método consiste na escolha das operações genéticas; é classificado pela escolha do tipo de operação genética a ser realizada em cada passo do ciclo de busca e pela manipulação dos alelos cromossômicos em função da forma de atuação dos operadores genéticos. O segundo método relaciona-se com o procedimento para escolha de indivíduos que sofrem ações dos operadores genéticos. O terceiro método relaciona-se com a escolha da quantidade de indivíduos para compor a população permanente, após o término de cada *ciclo de busca*, e constitui a fase final do algoritmo de *fitness*; a quantidade de indivíduos a ser incluída varia entre 1 e o número máximo de indivíduos que compõem a população permanente.

Ambas as categorias do processo de seleção utilizam o algoritmo de geração de números pseudo-aleatórios proposto em (Press *et al.*, 1994). A qualidade do gerador de números aleatórios influencia fortemente no desempenho do algoritmo genético.

3.5 Modelo AG-Paralelo

Um algoritmo genético paralelo (AGP) para alocar autoestruturas através de soluções do problema RLQ, chamado $AGP\text{-}RLQ$ foi desenvolvido e implementado tendo por base o modelamento das matrizes de ponderação Q e R, subseção (3.4.1), as operações genéticas, subseção (3.4.2), e a função de *fitness*, subseção (3.4.2). O modelo do algoritmo genético paralelo utilizado baseia-se na paralelização de processos independentes, (Koza, 1992), onde cada processador da rede evolui a mesma população inicial e o paradigma de programação paralela utilizado é o $SPMD$ (procedimento-único dados-múltiplos), onde cada processador da rede executa o mesmo programa e pode processar diferentes conjuntos de dados.

A característica de distribuir o mesmo programa entre vários processadores da rede torna o paradigma $SPMD$ bastante adequado para implementações do algoritmo $AGP\text{-}RLQ$, porque eles estão encarregados do processamento de soluções sequenciais do problema RLQ e da comunicação entre processadores.

A Figura (3.5) ajuda na melhor compreensão do modelo de paralelização proposto. Um processador mestre, ou coordenador, e diversos escravos ou coordenados são considerados os macro-elementos deste paradigma, porque os processamentos RLQ e a comunicação entre processadores são caracterizáveis por estes tipos de elementos. O coordenador tem as seguintes atri-

buições por ordem de execução: criar tarefas coordenadas, gerar e distribuir a população inicial para as tarefas coordenadas, realizar buscas, receber resultados RLQ e selecionar os melhores controladores. Cada coordenado (escravo) recebe a população inicial, resolve um problema RLQ sequencialmente e envia o resultado para o coordenador. O processador mestre comunica-se com todos os processadores escravos e estes só se comunicam com o processador mestre; só existem dois modos de comunicação: mestre-escravos e escravos-mestre no início e no final da busca, respectivamente.

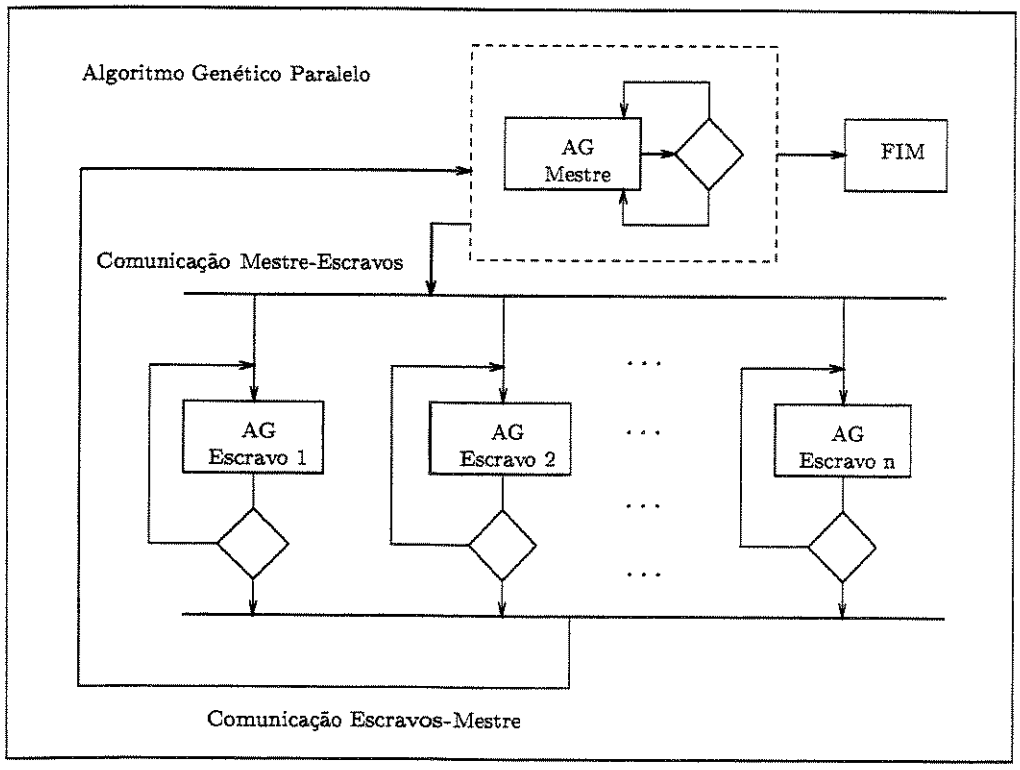

Figura 3.5: Modelo do Otimizador Genético Paralelo.

A finalização do algoritmo acontece quando os valores de Δ_{atual}, equação (3.16) e E_{atual}, equação (3.17), de cada tarefa não podem ser mais melhorados ou a autoestrutura especificada não pode ser obtida para um número máximo de passos estabelecido para o *ciclo de busca.*

Tendo como base as taxonomias apresentadas na seção (3.3), apresenta-se

a classificação do otimizador genético proposto. Sobre a granularidade, a taxa de processamento/comunicação é altíssima porque só existe comunicação entre processadores no início e no final de cada busca. Logo, o algoritmo pode ser considerado como sendo de granularidade grossa. Ele, também, pode ser considerado como sendo um AG hierárquico porque no alto nível possui múltiplas populações, contudo na primeira iteração do *ciclo de busca* a população é a mesma para todos os processadores, e no nível inferior as populações são únicas. O algoritmo não pode ser considerado global porque durante as operações genéticas cromossômicas somente parte da população permanente participa desta fase. Uma característica nitidamente marcante do algoritmo proposto é que este baseia-se em processamentos independentes, isto é, todos os AG partem da mesma população inicial mas seguem direções distintas durante as repetições do *ciclo de busca*. Conclui-se que o algoritmo genético proposto é hierárquico, granularidade grossa e com processamentos independentes.

Capítulo 4

Estratégias de Busca

4.1 Introdução

A unidade de decisão (*UD*) é uma estrutura lógica, onde decisões são tomadas e estas decisões fundamentam-se em estratégias previamente formuladas que têm por base elementos da inteligência computacional. As estratégias têm por objetivo direcionar a busca do otimizador genético de uma forma inteligente, tendo como base o conhecimento do processo. As medidas das funções de custo, critério de otimalidade de *Pareto*, teoria do *schemata*, paradigma do *multiarmed bandit*, ajuste de parâmetros em algoritmos baseados em computação evolutiva, operadores booleanos e relacionais são os elementos utilizados para obtenção de valores que influenciam a tomada de decisões no sentido de direcionar a busca.

A seção (4.2) apresenta a unidade lógica de decisão como parte integrante e desacoplada do algoritmo genético multiobjetivo. Diz-se que esta unidade é integrante porque fornece melhorias na qualidade da busca do algoritmo genético e diz-se que ela é desacoplada porque o algoritmo genético consegue efetuar a busca sem a sua presença. Nesta seção salienta-se o objetivo da unidade lógica e como ela atua em conjunto com o otimizador genético.

O teorema do *schemata*, o paradigma do *multiarmed bandit*, o ajuste de parâmetros do *otimizador-AG*, o critério de otimalidade de *Pareto* e a articulação de preferências são os elementos utilizados para montar as estratégias da unidade de decisão e estas estratégias têm por objetivo direcionar o ciclo de busca. As características destes elementos e suas modelagens em termos do problema de alocação de autoestruturas são apresentados na seção (4.3).

A fim de melhor explorar o espaço de solução em relação às características dos indivíduos desenvolveu-se um conjunto de funções de *fitness* que é chamado de *time de funções de fitness*. Este time de funções simula no espaço de solução vários ambientes que são representações diferentes do mesmo objetivo e têm por finalidade capturar as características que uma única função de *fitness* não tem capacidade de representar devido à sua natureza estrutural. Os fundamentos e a modelagem do time são apresentadas na seção (4.4).

4.2 Unidade Lógica de Decisão

A unidade lógica de decisão (*ULD*) e o otimizador que tem por base algoritmos genéticos, capítulo (3), são os elementos básicos do algoritmo genético multiobjetivo (*AGMO*), Figura (4.1). O otimizador genético (*otimizador-AG*) encarrega-se da busca de indivíduos-QR e a *ULD* é uma unidade lógica de decisão projetada para guiar o otimizador no sentido de promover melhoramentos na busca, seja na qualidade da solução, seja no aumento da velocidade de resposta.

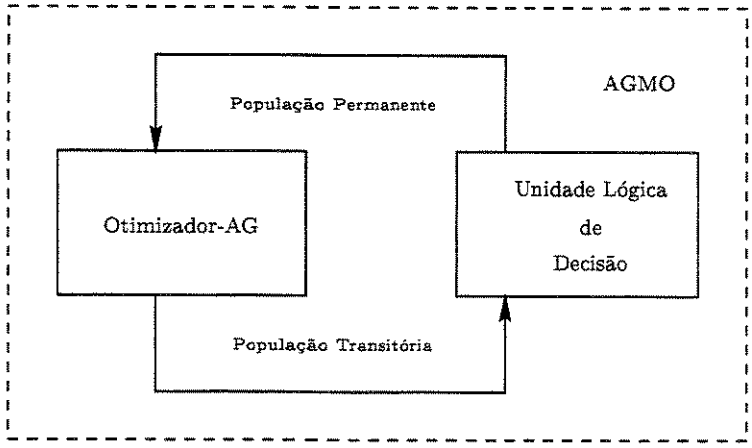

Figura 4.1: Estrutura Básica do Algoritmo Genético Multiobjetivo.

A estrutura do *AGMO* é constituída pelo par otimizador-AG e *ULD*, equação (4.1), que realiza explorações sequenciais no espaço de busca S.

$$AGMO = (Otimizador - AG, ULD) \tag{4.1}$$

A operação da unidade lógica de decisão pode ser caracterizada como tendo dois modos básicos de atuação para direcionar a busca. O primeiro modo tem por base o teorema do *schema*, o paradigma do *multiarmed bandit*, (Koza, 1992), e o ajuste de parâmetros de algoritmos baseados em computação evolutiva, (Eiben *et al.*, 1999). O segundo modo está fundamentado no critério de otimalidade de *Pareto*, na articulação de preferências, na indução de nichos e na restrição de *mating*, (Fonseca e Fleming, 1998), (Goldberg, 1989).

4.3 As Estratégias

As ações das estratégias de busca e os seus gatilhamentos dependem de medidas e de informações fornecidas pelas funções de *fitness* aos mecanismos que implementam as estratégias baseadas no critério de otimalidade de *Pareto*, teoria do *schemata*, paradigma do *multiarmed bandit* e ajuste de parâmetros. Por sua vez, estas medidas e informações são processadas através de operadores booleanos e relacionais, e os resultados são utilizados para direcionar a busca de forma inteligente, no sentido de aumentar sua velocidade e melhorar a qualidade da solução.

A seguir, descreve-se como estes mecanismos são construídos ou modelados para fornecer decisões que permitam uma melhor exploração do espaço de solução.

4.3.1 Estratégias do Schemata e *MAB*

Esta subseção apresenta as estratégias que têm por base a teoria do *schemata* e o paradigma do *multiarmed bandit* (*PMAB*). A teoria do *schemata* (*Tschema*) é utilizada para evitar a proliferação de determinados tipos de *schemata* que têm como consequência uma redução da perda de variedade genética da população permanente. O paradigma do *multiarmed bandit* tem como finalidade tentar extrair as potencialidades de interesse para a solução do problema que podem existir nos indivíduos com baixo grau de adequabilidade.

As ações destas estratégias dependem do gatilhamento de três regras. A primeira atua na deteção da convergência prematura da população ou do melhor indivíduo. A segunda atua de forma aleatória tendo por base uma pequena probabilidade de ocorrência. A terceira atua de maneira determinística

entre vida e morte de gerações, isto é, esta é disparada periodicamente de acordo com uma certa quantidade de iterações do *ciclo de busca.*

Os mecanismos de atuação destas estratégias são discutidos no nível de interações entre o *otimizador-AG* e a unidade de decisão, no nível de escolha de indivíduos e no nível de efetivação de operações genéticas.

As interações entre a operação genética *crossover* e as estratégias *Tschema* e *PMAB*, salientando a funcionalidade dos elementos que compõem o mecanismo de atuação destas estratégias, são explicadas tendo como referência a Figura (4.2). Estas estratégias têm por base escolhas aleatórias de indivíduos e decisões, gatilhamentos de regras e dois bancos de dados de indivíduos-QR que são montados durante repetições do *ciclo de busca.* Um banco de dados é montado só com *schemata* factíveis, chamado *schemata-F*, que são os indivíduos que apresentaram os melhores *fitness* durante as buscas. O segundo banco de dados, chamado *schemata-NF*, é montado com os indivíduos não factíveis e sua formação é similar à do banco de dados factível.

Estas estratégias atuam integradas com a operação *crossover*, Figura (4.2), no sentido de que esta operação genética artificial executa uma das estratégias. Todas as vezes, antes da efetuação da operação *X-over*, a unidade de decisão verifica a ocorrência de três eventos e estes eventos são definidos como regras de gatilhamento. Se a ocorrência de um destes eventos é detectada, uma escolha aleatória é executada; a aleatoriedade da escolha é representada por uma roleta, para o tipo de indivíduo, *schemata-F* ou *schemata-NF*, que é combinado com um indivíduo escolhido aleatoriamente da população permanente. Após a definição do tipo de indivíduo, o próximo passo é retirar aleatoriamente um indivíduo do respectivo banco de dados. O último passo é enviar o indivíduo escolhido para ser combinado no *otimizador-AG.*

As interações entre a estratégia e o *otimizador-AG* estão representadas com mais detalhes na Figura (4.3), isto é, no nível de como a escolha de indivíduos dos bancos de dados e a operação *crossover* são realizadas dentro da estrutura do algoritmo genético multiobjetivo. Depois de definir uma estratégia do tipo *Tschema* ou *PMAB*, Figura (4.2), duas escolhas aleatórias são feitas antes de enviar o indivíduo para o *otimizador-AG*, Figura (4.3).

Considerando que a estratégia *Tschema* foi a definida pela escolha aleatória, Figura (4.3), um indivíduo-QR é retirado do banco de dados factível, utilizando uma roleta igualmente ponderada. Depois desta escolha, uma roleta bipolar é utilizada para definir o tipo de schema do indivíduo-QR que será explorado durante a operação de *crossover*; no caso o tipo escolhido foi *schema-Q*. O indivíduo escolhido é enviado para o *otimizador-AG* e os

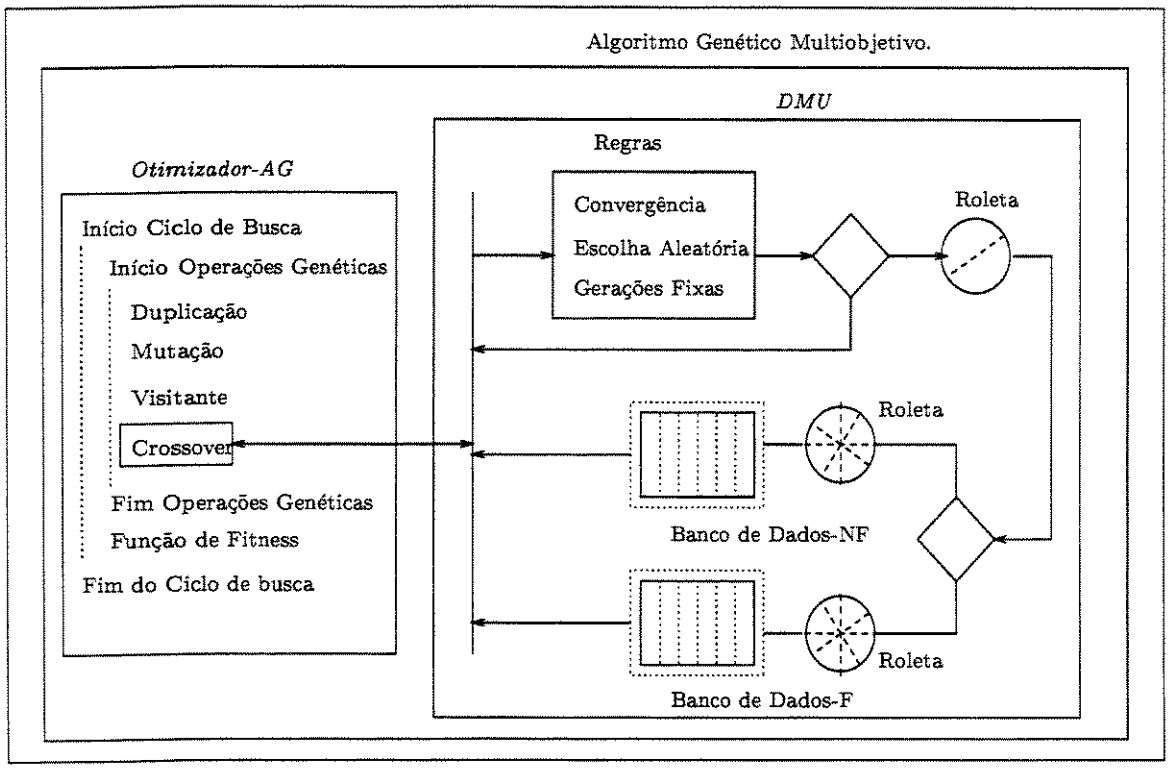

Figura 4.2: Ponto de interação entre o *Otimizador-AG* e as Estratégias baseadas no *Tschema* e *PMAB*.

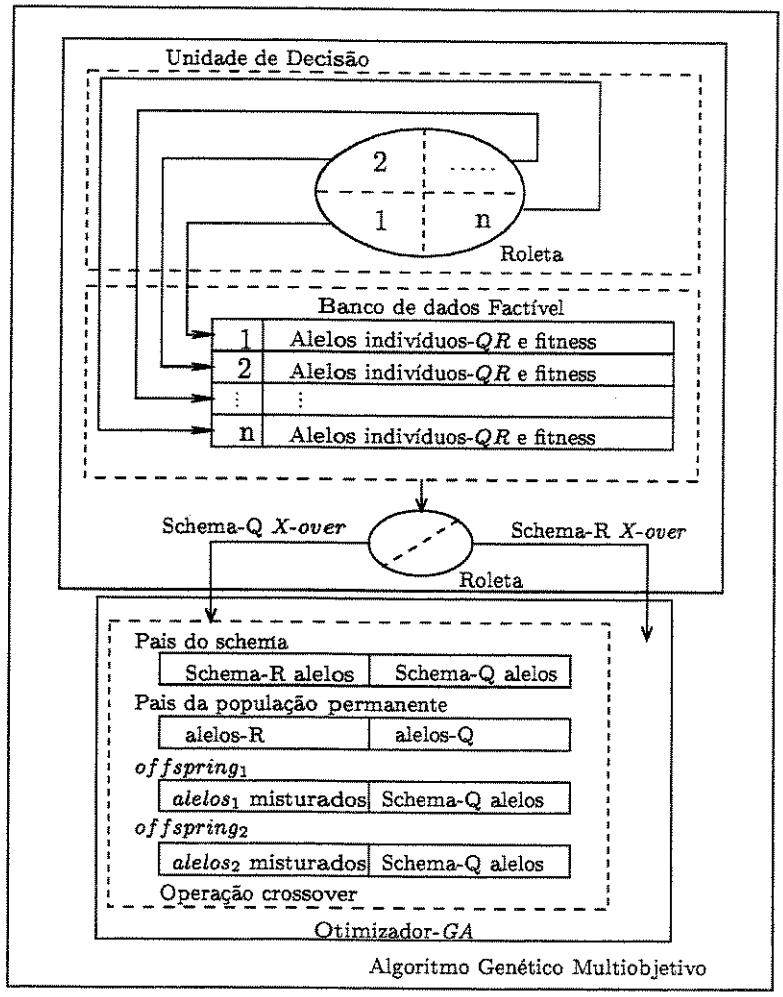

Figura 4.3: Estratégias do Schemata e *MAB* - Escolha do indivíduo schema e a operação *crossover*.

resultados da operação *crossover* são dois *off-springs* com os mesmos alelos-Q e dois alelos-R diferentes misturados, que são obtidos combinando os alelos-R do indivíduo do banco de dados factível e os alelos R de um indivíduo da população permanente, escolhido aleatoriamente através de uma roleta ponderada pelo valor de *fitness*.

4.3.2 Indução de Nichos

A indução de nichos é estabelecida de forma natural com o desenrolar do processo de busca. Se os indivíduos-QR encontrarem uma dada região onde existe a tendência a fortalecer as características genéticas artificiais de sua espécie, estes tendem a permanecer nesta região que é chamada de nicho. Nesta tese a palavra nicho tem dupla conotação, no sentido de que os nichos são induzidos de formas artificial e natural.

Uma indução artificial de nichos significa que indivíduos-QR são deslocados para uma determinada região e que estão fadados a permanecer dentro desta região até o final das iterações do *ciclo de busca*. Uma indução natural de nichos significa que certos indivíduos estão se fortalecendo, aumentando espontaneamente o nível de adequabilidade, de propriedades de uma determinada região do espaço de solução.

A modalidade artificial é o tipo de método para indução de nichos desenvolvido nesta tese. O método desenvolvido tem como fundamento o seguinte princípio: a indução artificial de nichos tem como base o deslocamento dos indivíduos da população inicial para regiões específicas do espaço de busca e critérios de restrição de *mating* são utilizados para guiar a exploração deste nicho. Este princípio está harmoniosamente casado com a característica de paralelismo da estrutura de algoritmo genético multiobjetivo, desenvolvido para a alocação de autoestruturas através do projeto RLQ. Com esta situação, cada processador tem por objetivo a exploração de uma dada região do espaço de busca, isto é, os processadores exploram nichos artificiais. Os próximos parágrafos têm por objetivo formalizar o método para indução artificial de nichos.

O espaço S de busca é repartido em S_k regiões. Uma delas é explorada pela população inicial e as outras $k-1$ são exploradas por uma nova população que é construída a partir de um incremento dado a cada indivíduo-QR da população inicial. A linha de fronteira entre regiões é definida por todos os indivíduos-QR menos a metade do incremento dado a todos os indivíduos da população inicial; desta maneira a população inicial é deslocada para o

centro do nicho artificial.

Considerando o espaço S de busca como sendo o conjunto de todos os indivíduos-QR cujos alelos estão em uma faixa especificada, o espaço de busca completo é definido como:

$$S = \{QR - individuos \mid M_{inf} \leq QR_{alelo_i} \leq M_{sup}\} \quad (4.2)$$
$$i = 1, \ldots, t_{pop}(m^2 + n^2)$$

onde QR_{alelo_i} representa o *i-ésimo* alelo da população, M_{inf} e M_{sup} são o maior e o menor valor, respectivamente, que um alelo pode atingir, m é a quantidade de alelos Q, n é a quantidade de alelos R e t_{pop} é o tamanho da população permanente.

O espaço de busca total S pode ser representado como a união de todas as regiões S_i, $i = 1 \ldots k$. Então:

$$S = \{S_1 \cup S_2 \cup S_3 \cup \ldots S_k\} \quad (4.3)$$

A seguir, tendo como base (4.2), formaliza-se as S_k regiões. Inicialmente, considera-se que a região S_1 é dada por:

$$S_1 = \{X_i \mid M_{inf} \leq QR_{alelo_i} \leq X_i + \Delta_l\} \quad (4.4)$$

onde X_i representa a população inicial, M_{inf} e $X_i + \Delta_l$ representam os limites inferior e superior que cada alelo pode assumir na região S_1 e Δ_l é o incremento base que é utilizado para estabelecer fronteiras e deslocar a população inicial.

As regiões restantes são generalizadas da seguinte maneira:

$$S_k = \{X_i + k\Delta_l - \Delta_l/2 \mid \quad (4.5)$$
$$X_i + (k-1)\Delta_l \leq QR_{alelo_i} \leq X_i + k\Delta_l\}$$
$$i = 1, \ldots, t_{pop}(n^2 + m^2); \qquad k = 2, \ldots, n_{reg}$$

onde k é um índice que referencia à k-ésima região, $k\Delta_l - \Delta_l/2$ representa a posição inicial da população X_i em cada região k, $X_i + (k-1)\Delta_l \leq QR_{alelo_i} \leq X_i + k\Delta_l$ representa as fronteiras entre as regiões do plano-QR e n_{reg} é a quantidade de regiões do espaço de solução. A fronteira entre a última região,

chamada de S_l, e as regiões onde não existirá busca, S_{l+1} é dada por M_{sup}, isto é, o *otimizador-AG* não realiza buscas em regiões cujos alelos satisfazem a relação $QR_{alelo_i} > M_{sup}$.

Uma vez estabelecidas S_k fronteiras, o *otimizador-AG* inicia o *ciclo de busca*. Antes do otimizador iniciar cada busca, a unidade de decisão procura grupos de indivíduos que possuem características similares e que têm seus valores de função de *fitness* dentro de uma faixa definida; isto é, esta procura significa uma tentativa de estabelecer nichos naturais que estão contidos na *k-ésima* região S_k. Então, qualquer nicho S_{ki}^{nicho} na região S_k satisfaz a relação:

$$S_{ki}^{nicho} \subset S_k, \qquad ki = 1, \ldots, k_{nich} \tag{4.6}$$

onde k_{nich} é a quantidade de nichos na região S_k.

As condições de satisfabilidade para o estabelecimento de nichos nas regiões S_k são definidas pela relações dadas por:

$$M_{alelo_j}^{inf} \leq alelos - QR_j^{nicho} \leq M_{alelo_j}^{sup}, \qquad j = 1, \ldots, n_{nich} \tag{4.7}$$

onde $alelos - QR_j^{nicho}$ representa os alelos do *j-ésimo* indivíduo do nicho e $M_{alelo_j}^{inf}$ e $M_{alelo_j}^{sup}$ representam os limites inferior e superior de cada alelo do *j-ésimo* indivíduo, respectivamente. A quantidade de indivíduos do nicho é n_{nich}.

A condição necessária para um indivíduo-QR ser aceito na população permanente de um nicho é dada por:

$$fitness(QR_j^{nicho}) \leq fitness_{lim}^{nicho}, \qquad j = 1, \ldots, n_{nich} \tag{4.8}$$

onde $fitness(QR_j^{nicho})$ representa o valor de fitness do *j-ésimo* indivíduo do nicho e $fitness_{lim}^{nicho}$ representa o limite do valor de *fitness* que o *j-ésimo* indivíduo deve satisfazer.

Quando a *UD* detecta estas características, significa que está acontecendo a formação de um nicho natural: o próximo passo é o estabelecimento do tamanho do nicho e dos limites do critério de *mating* ou acasalamento para iniciar a sua exploração.

O tamanho do nicho natural: suas fronteiras são estabelecidas a partir dos alelos dos indivíduos-QR mais distantes do núcleo do nicho, sendo necessários no mínimo três alelos para delimitar a sua fronteira; os alelos mais distantes

do núcleo são incrementados de um valor no sentido oposto ao sentido do núcleo. Então, conclui-se que para cada alelo da população do nicho existe um nicho-alelo. A equação (4.9) representa o estabelecimento da fronteira de um nicho natural.

$$\begin{aligned} alelo - QR_{ij}^{fnicho} &= alelo - QR_{ij}^{nicho} + \Delta^{nicho} \\ & i = 1, \ldots, n^2 + m^2, \qquad j = 1, \ldots, n_{nich} \end{aligned} \tag{4.9}$$

onde $alelo - QR_{ij}^{fnicho}$ estabelece a fronteira do nicho natural e Δ^{nicho} é o incremento dado aos alelos mais distantes do núcleo do nicho.

A restrição de *mating* segue o critério de proximidade de alelos:

$$|(alelo - QR_{ij}^{nicho}) - (alelo - QR_{ij\pm\xi}^{nicho})| \leq M_{mating}^{nicho} \tag{4.10}$$

onde $alelo - QR_{ij}^{nicho}$ e $alelo - QR_{ij\pm\xi}^{nicho}$ representam os *i-ésimos* alelos do *j-ésimo* e o do $j \pm \xi$-*ésimo* indivíduos-QR, $j \pm \xi$ representa qualquer indivíduo do nicho tal que $j \pm \xi \neq j$, que são candidatos a participarem do *mating pool*, respectivamente. A diferença entre os *i-ésimos* alelos, no mínimo 80%, de dois indivíduos deve satisfazer a restrição (4.10) a fim de garantir a ocorrência de acasalamento.

O otimizador-AG efetua a exploração do nicho. A unidade de decisão monitora a fronteira do nicho, garantindo que operações genéticas cromossômicas sejam realizadas somente com indivíduos que pertencem ao nicho; a satisfabilidade do critério de *mating*, garante que indivíduos com características semelhantes ou muito próximas não participem do mesmo *mating pool.* Como consequência, tem-se a não redução da diversidade do material genético dos indivíduos do nicho.

4.3.3 O Critério de Otimalidade de Pareto

O conceito de otimalidade de *Pareto* é utilizado para garantir que o próximo vetor de solução factível seja melhor do que a última solução factível; isto é, a próxima solução é considerada uma solução melhor se os valores de todos os componentes desta estão mais próximos do conjunto de restrições do que os valores correspondentes do vetor solução anterior.

Seja C_{RS}^{k} o conjunto que representa as sensibilidades dos autovalores, calculadas na *k-ésima* iteração do *ciclo de busca* que é dado por:

$$C_{RS}^{k} = \{s_1^k(Q,R), s_2^k(Q,R), \ldots, s_n^k(Q,R)\} \tag{4.11}$$

onde $s_1^k(Q,R), s_2^k(Q,R), \ldots, s_n^k(Q,R)$ representam os n valores numéricos das sensibilidades dos autovalores na iteração k como uma função das matrizes de ponderação Q e R.

A busca efetuada na iteração k é considerada melhor solução do que a solução da iteração $k-1$ se e somente se as seguintes relações forem satisfeitas:

$$\begin{aligned} s_1^k(Q,R) &< s_1^{k-1}(Q,R) && (4.12)\\ s_2^k(Q,R) &< s_2^{k-1}(Q,R) && (4.13)\\ &\vdots \quad \vdots \quad \vdots \\ s_n^k(Q,R) &< s_n^{k-1}(Q,R) && (4.14) \end{aligned}$$

onde $s_1^{k-1}, s_2^{k-1}, \ldots, s_n^{k-1}$ representam as n sensibilidades dos autovalores na iteração $k-1$.

O conjunto de desigualdades (4.12-4.14) é verificado a cada iteração do *ciclo de busca* até que o vetor de restrições seja satisfeito. As desigualdades podem ser relaxadas, quando uma ou mais restrições são satisfeitas e as restantes são iguais.

4.3.4 Articulação de Preferências

O conceito de articulação de preferências é utilizado para direcionar a busca dos indivíduos-QR de forma a satisfazer a restrição mais difícil, e quando esta restrição é satisfeita a unidade de decisão autoriza o *otimizador-AG* a determinar a solução que satisfaz a segunda restrição mais difícil, e o processo continua desta maneira até satisfazer todos os componentes de um dado vetor de restrições preferenciais.

O critério de articulação de preferências tem prioridade sobre o critério de otimalidade de *Pareto*. Somente após a satisfação de certas restrições que são consideradas preferenciais é que o critério de otimalidade de *Pareto* é ativado pela unidade de decisão.

4.3.5 Ajuste de Parâmetros

A adaptação de parâmetros em algoritmos de computação evolutiva (ACE) não é uma tarefa fácil, porque existe um boa quantidade de parâmetros

que podem ser ajustados e a coordenação entre estes parâmetros deve ser levada em consideração a fim de se obter um ajuste efetivo. Os métodos desenvolvidos nesta tese, seguem a taxonomia de ajuste de parâmetros em *ACE* proposto por (Eiben *et al.*, 1999), e no intuito de conceituar certas estratégias propõe-se uma extensão de certos termos da referida taxonomia.

A Tabela (4.1) apresenta os parâmetros do otimizador-AG que podem ser ajustados de forma a melhorar o desempenho da busca. Nesta aplicação, o ajuste do parâmetro idade da operação *crossover* foi escolhido para ser controlado durante as iterações do *ciclo de busca*, porque a busca é intensificada sobre a operação cromossômica *crossover*.

Classe	Função	Parâmetro
Ciclo de Busca	Probabilidade de *X-over*	P_X
	Probabilidade de mutação	P_M
	Probabilidade de duplicação	P_D
	Probabilidade de visitante	P_V
Operações Genéticas	Fator Q % *X-over*	q_{idad}
	Fator R % *X-over*	r_{idad}
	Fator de mutação global	FG_M
	Base de mutação local	LB_M
	Expoente de mutação local	LE_M
População	Tamanho	Pop_S
	Quantidade de indivíduos substituídos em cada geração	Pop_R

Tabela 4.1: Classificação dos Parâmetros do *otimizador-AG*.

A idéia geral na qual o método proposto baseia-se para regular a intensidade da variação dos parâmetros da operação *X-over* está ilustrada na Figura (4.4). O espaço de busca é dividido em Q quadrantes e estes são chamados de nichos artificiais, conforme definido na subseção (4.3.2). A idéia principal consiste em deslocar alguns indivíduos-QR para outros quadrantes do espaço de busca ou mesmo mover indivíduos em regiões do mesmo quadrante através de variações dos parâmetros r_{idad} ou q_{idad} da operação de *X-over*, e explorar áreas do quadrante através de pequenas variações nestes parâmetros, isto é, transformando o operador *crossover* em mecanismo de busca local. A quantidade de iterações do *ciclo de busca* em uma dada região é estabelecida pela quantidade de tentativas que não foram bem sucedidas.

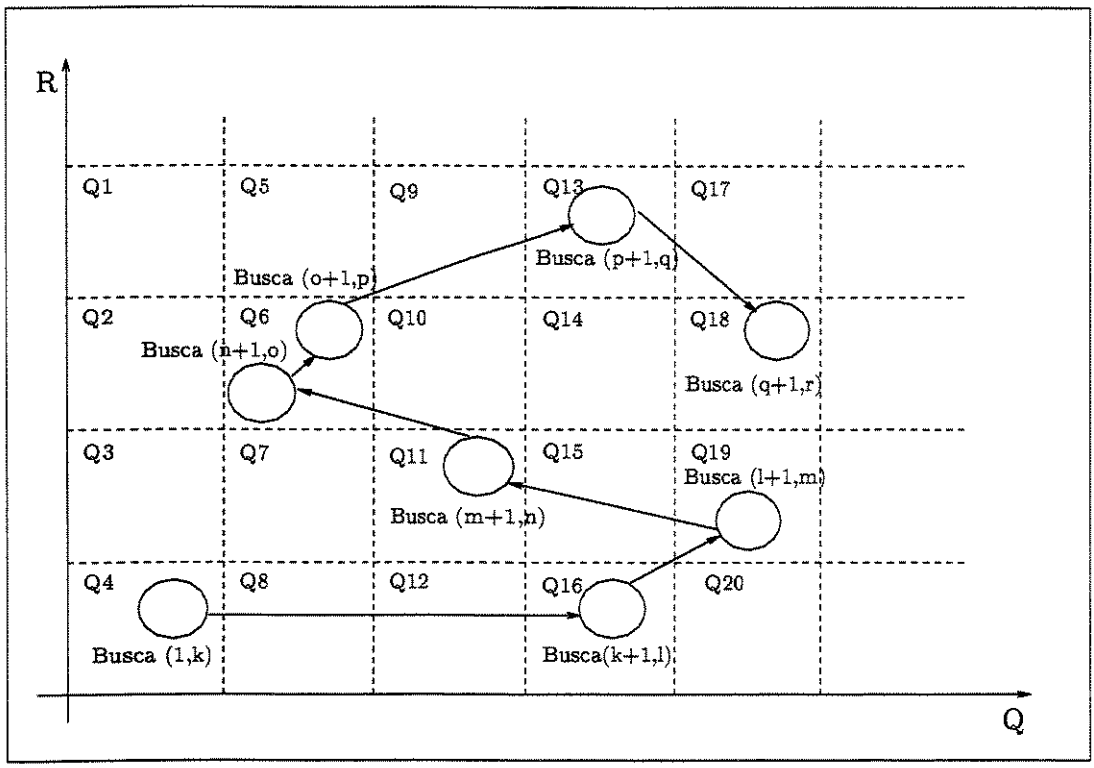

Figura 4.4: Exploração do espaço de busca via variações nos parâmetros da operação *X-over*.

O otimizador-AG proposto realiza três tipos de operação *X-over*. O primeiro tipo é a operação padrão, os indivíduos-QR filhos são formados das partes de dois pais selecionados aleatoriamente, *fitness proporcional*. O segundo tipo tem por base a teoria do *schemata* e um banco de dados montado com os melhores indivíduos, todos eles factíveis, resultantes das iterações do ciclo de busca; um dos pais vem da população permanente, escolhido de maneira idêntica aos indivíduos da operação padrão, e o outro é proveniente do banco de dados factível, escolhido aleatoriamente de acordo com a sua posição de armazenamento; após a seleção de indivíduos, outra escolha é realizada para definir o tipo de *schema* que será mantido após a operação de *crossover*. O terceiro tipo de operação é realizado de maneira similar ao segundo tipo, contudo um dos pais é escolhido de um banco de dados montado

com indivíduos não factíveis; esta operação tem em vista a exploração do paradigma do *multiarmed bandit*. Estas operações encontram-se bem definidas e comentadas no capítulo (3), subseção (3.4.2). Contudo, para uma melhor compreensão da estratégia de ajuste de parâmetros, comenta-se novamente aspectos importantes destes modelos de operações cromossômicas *X-over*.

O primeiro tipo de ajuste de parâmetros é dado pelas seguintes relações:

$$\begin{aligned} QR_{filho_1} &= Q_{filho_1} \cup R_{filho_1} \\ QR_{filho_2} &= Q_{filho_2} \cup R_{filho_2} \end{aligned} \tag{4.15}$$

onde QR_{filho_1} e QR_{filho_2} são os filhos resultantes da combinação linear dos alelos Q dos pais, equações (4.16):

$$\begin{aligned} Q_{filho_1} &= q(t)_{idad} Q_{pai_1} + |q(t)_{idad} - 1| Q_{pai_2} \\ Q_{filho_2} &= q(t)_{idad} Q_{pai_2} + |q(t)_{idad} - 1| Q_{pai_1} \end{aligned} \tag{4.16}$$

onde $q(t)_{idad}$ é o parâmetro da operação *X-over* dos alelos Q que são modificados durante as iterações do *ciclo de busca* de acordo com regras definidas pelo projetista. Q_{pai_1} e Q_{pai_2} são os alelos-Q dos indivíduos QR_{pai_1} e QR_{pai_2}, respectivamente. Os alelos R_{filho_1} e R_{filho_2}, resultantes de operação *X-over* sobre os cromossomos-R, são montados de maneira similar aos alelos Q.

Os parâmetros $q(t)_{idad}$ ou $r(t)_{idad}$ para tipo 2 e 3 de operação *crossover* podem assumir o valor 1 (*um*) no *ciclo de busca* em consideração e o estabelecimento do valor unitário para somente um dos parâmetros depende da escolha do tipo de *schemata-Q* ou *schemata-R*. Considerando que um *schemata-Q* foi escolhido aleatoriamente do banco de dados MAB, a equação (4.17) representa os dois filhos que são formados a partir de alelos $Q_{pai_{MAB}}$ do pai e o parâmetro $q(t)_{idad}$ assume o valor 1.

$$\begin{aligned} Q_{filho_1} &= Q_{pai_{MAB}} \\ Q_{filho_2} &= Q_{pai_{MAB}} \end{aligned} \tag{4.17}$$

onde $Q_{pai_{MAB}}$ são os alelos-Q provenientes do banco de dados MAB.

Os alelos R_{filho_1} e R_{filho_2} são montados como uma combinação linear dos alelos R_{pai_1} e $R_{pai_{MAB}}$. Os cromossomos-Q filho estão representados pela equação (4.18).

$$\begin{array}{rcl} R_{filho_1} & = & r(t)_{idad} R_{pai_1} \quad + \quad |r(t)_{idad} - 1| R_{pai_{MAB}} \\ R_{filho_2} & = & r(t)_{idad} R_{pai_{MAB}} \quad + \quad |r(t)_{idad} - 1| R_{pai_1} \end{array} \tag{4.18}$$

onde $r(t)_{idad}$ é o parâmetro da operação cromossômica que pode ser ajustado a cada iteração do *ciclo de busca* correspondente aos alelos do tipo R.

O modelo que representa as variações dos parâmetros da operação *crossover*, para o caso específico do parâmetro q_{idad}, e define o grau de combinação entre os alelos de dois indivíduos-QR, é dado pela equação:

$$q(t)_{idad} = \begin{cases} q(t-1)_{dir} + \Delta q(t), & \text{Se } q_{sent} > 0 \\ q(t-1)_{esq} - \Delta q(t), & \text{Se } q_{sent} < 0 \end{cases} \tag{4.19}$$

onde q_{sent} é a variável que define o sentido da variação do parâmetro $q(t)_{idad}$; quando $q_{sent} > 0$ o parâmetro sofre variações à direita, $q(t-1)_{dir}$, e quando $q_{sent} < 0$ as variações ocorrem à esquerda, $q(t-1)_{esq}$. $\Delta q(t)$ representa o passo de variação de $q(t)_{idad}$.

O parâmetro $q(t)_{idad}$, equação (4.19), varia ao longo de uma linha reta e o seu sentido é definido pela média da função de *fitness* (sensibilidades e função de custo), a melhor solução factível e por intensificações de buscas em um determinado sentido.

Os valores de $q(t)_{idad}$ e $\Delta q(t)$ são reinicializados deterministicamente durante intervalos sincronizados após a ocorrência de um determinado número de iterações do *ciclo de busca*, que pode ser definido pelo projetista. A Figura (4.5) ilustra graficamente os pontos de reinicialização destes dois parâmetros.

O sentido da variação do parâmetro $q(t)_{idad}$ é definido pela função q_{sent} que é uma função *booleana*, equação (4.20). Quando está função assume um valor maior que 0 (*zero*), a variação do parâmetro $q(t)_{idad}$ segue para a direita e quando assume um valor menor que 0 (*zero*) a variação de $q(t)_{idad}$ segue para a esquerda. Então, a função booleana:

$$q_{sent} = q_d(\Delta(Q,R)^{med}, E(Q,R)^{med}, \Delta(Q,R)^{max}, \Delta D_q) \tag{4.20}$$

onde $\Delta(Q,R)^{med}$ é a média das piores sensibilidades: cada indivíduo da população permanente contribui com a sua pior sensibilidade e a média destas é dada por $(\sum_j^N \max_i s_{ji}^t(Q,R) \ /N)$, onde N é o tamanho da população, na iteração t do *ciclo de busca*. $E(Q,R)^{med}$ é o valor médio da função de custo da estrutura de *fitness* da população, $(\sum_i^N E_i^t(Q,R)/N)$, na iteração t.

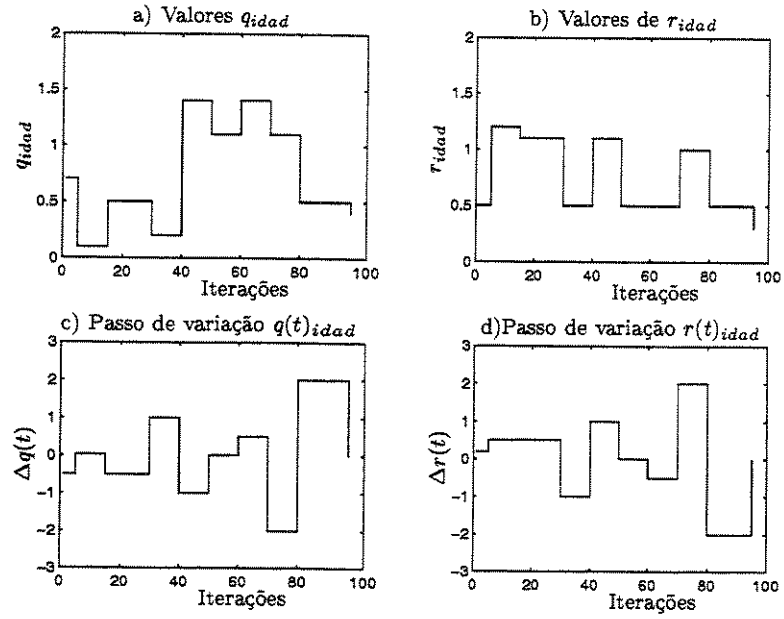

Figura 4.5: Reinicialização dos parâmetros da operação *crossover* e seus passos de variação *versus* iterações do ciclo de busca.

$\Delta(Q,R)^{max}$ é dado por $\min_j \max_i s(Q,R)_i^t$ e representa o melhor indivíduo da população no passo t. ΔD_q é o número máximo de variações permitidas em um dado sentido.

A função booleana q_{sent}, equação (4.20), representada por uma expressão em termos de operadores lógicos e relacionais, é dada pela equação (4.21). A função q_{sent} impõe um novo sentido para as variações do parâmetro $q(t)_{idad}$, equação (4.19), se e somente se todas as suas cláusulas são falsas. A referida expressão é dada por:

$$q_{sent} = \delta(t)^{med} \vee e(t)^{med} \vee \delta(t)^{max} \vee \delta D(t)_q \qquad (4.21)$$

onde $\vee$ representa o operador booleano *ou*, $\delta^{med}(t)$ é uma função booleana cujo resultado é proveniente da comparação $(\Delta(Q,R,t)^{med} < \Delta(Q,R,t-1)^{med})$; $e(t)^{med}$ assume um valor binário resultante da comparação $(E(Q,R,t)^{med} < E(Q,R,t-1)^{med})$; $\delta(t)$ é uma função booleana cuja expressão é montada a partir da relação entre as piores sensibilidades

Método	Regra de Variação	$\Delta q(t)$	$\Delta r(t)$	ΔD_Q	ΔD_R
Quase-dinâmica	Vários pontos do Ciclo de Busca	0	0	0	0
Dinâmica	Adaptativa	$\neq 0$	$\neq 0$	$<< CB$	$<< CB$
	Determinística	$\neq 0$	$\neq 0$	∞	∞

Tabela 4.2: Técnicas para Variações dos Parâmetros.

nas iterações t e $t-1$, respectivamente, $(\Delta(Q,R,t)^{max}$ ¡ $\Delta(Q,R,t-1)^{max})$; $\delta D(t)$ é um valor binário resultante da comparação $D(t) < D_{val_q}$, onde D_{val_q} é a quantidade máxima de variações permitidas para o parâmetro em uma dado sentido, após uma quantidade de tentativas mal sucedidas.

O modelo de variação dos parâmetros, equação (4.19), permite a obtenção de três técnicas para determinar o grau de combinação dos cromossomos; o parâmetro $q(t)_{idad}$ regula a combinação dos alelos-Q e $r(t)_{idad}$ regula o grau de combinação dos alelos-R. A Tabela (4.2) apresenta estas técnicas; suas denominações são uma extensão da taxonomia de variações de parâmetros proposta por (Eiben *et al.*, 1999) e, neste contexto, referem-se ao ajuste de parâmetros dos modelos das operações *crossover* via alfabeto numérico decimal.

A primeira técnica é chamada de *quase*-dinâmica porque os parâmetros sofrem variações somente após a ocorrência de certa quantidade de iterações do *ciclo de busca* (CB); esta técnica é bem ilustrada pela Figuras (4.5a e b) e para esta situação os parâmetros Δq e Δr são nulos para todas as iterações do *ciclo de busca*. A segunda técnica é classificada como controlada de forma adaptativa porque os parâmetros Δq ou Δr e o parâmetro ΔD da expressão booleana (4.21) são menores do que o número total de iterações do *ciclo de busca*. A terceira técnica é deterministicamente controlada, seu parâmetro fixo ΔD_{valq} é sempre maior do que o número de iterações do *ciclo de busca* e o parâmetro que define o tamanho do passo de variação é não nulo; nesta técnica, os sentidos das variações dos parâmetros $q(t)_{idad}$ e $r(t)_{idad}$ são únicos, ou seja, a expressão booleana, equação (4.21), é constante para todas as iterações do *ciclo de busca* e o seu valor é definido pelo projetista.

4.4 Time de Funções de Fitness

Certos problemas podem ser decompostos em problemas menores e cada um destes problemas menores pode ser simultaneamente resolvido por diferentes métodos. A união das soluções dos subproblemas fornece a solução completa do problema total. O conjunto de diferentes métodos, atuando de forma paralela na solução de cada subproblema, constitui o que chamamos de *time de algoritmos*. A eficácia da aplicação desta técnica pode ser verificada na solução de problemas de sistemas de potência, (Báran *et al.*, 1996), e de jogos dinâmicos, (Costa Filho, 1992).

Nesta tese, propõe-se a utilização do conceito de *time de algoritmos* para o contexto dos algoritmos genéticos com o intuito de inserir ambientes com diferentes características no espaço de solução; com isto, espera-se aumentar o nível de adaptabilidade de certos indivíduos. Estes novos ambientes são diretamente representados por funções de fitness e o conjunto constituído por estes ambientes é chamado *time de funções de fitness* (*TFF*).

Um *time de funções de fitness* é definido como sendo um conjunto de funções de *fitness*:

$$TF_{fitness}^{n_{fit}} = ff_i, \qquad i = 1, \ldots, n_{fit} \tag{4.22}$$

onde ff_i representa a *i-ésima* função de *fitness* do time e n_{fit} é a quantidade de funções do time.

As funções do conjunto (4.22) podem competir entre si para explorar todo o espaço de busca ou podem ser alocadas para explorar regiões selecionadas. Estas regiões podem ser estabelecidas utilizando os métodos para formação de nichos propostos na subseção (4.3.2).

Funções de custo e critérios prévios de seleção são os dois elementos que definem cada função do time. Os critérios prévios de seleção são utilizados em uma primeira pontuação dos indivíduos da população transitória e as funções de custo podem ser utilizadas como uma segunda pontuação, seja para o caso de empate na pontuação ou para atuarem como elementos decisivos na seleção de indivíduos. A forma da atuação de cada função do time depende dos aspectos construtivos de cada função; devido à complexidade em relação ao aspecto construtivo, cada função do time é denominada de estrutura de *fitness*.

Uma estrutura do *time de funções de fitness* pode ser formalizada por:

$$ff_i = \{fc_i, cps_i\}, \qquad i = 1, \ldots, n_{fit} \tag{4.23}$$

onde fc_i é a *i-ésima* função de custo do time e cps_i representa o *i-ésimo* critério de seleção do time.

As restrições dos critérios prévios de seleção são as principais diferenças entre os elementos que constituem cada estrutura do time. A quantidade de restrições, o tipo e a natureza da função de custo podem tornar a busca mais difícil.

Cinco tipos de estruturas de função de *fitness* foram desenvolvidos nesta pesquisa. Contudo, somente duas funções de custo constituem o núcleo do *TFF*. Uma destas funções é a soma das sensibilidade dos autovalores, equação (2.105), chamada de J_S. A outra função de custo tem por base o erro quadrático da diferença entre a autoestrutura especificada e a calculada durante o processo de busca, equação (4.24), é chamada de J_E e foi utilizada por (Davis e Clarke, 1995):

$$\sum_{i=1}^{n} f_{\lambda i}(\lambda_{ei} - \lambda_{ci})^*(\lambda_{ei} - \lambda_{ci}) + (\vec{v}_{ei} - \vec{v}_{ci})^* F_{vi}(\vec{v}_{ei} - \vec{v}_{ci}) \tag{4.24}$$

onde λ_{ei} é o *i-ésimo* autovalor especificado, λ_{ci} é o *i-ésimo* autovalor calculado, $f\lambda_i$ é a ponderação do *i-ésimo* autovalor, $\vec{v}_{ei}$ é o *i-ésimo* autovetor especificado, $\vec{v}_{ci}$ é o *i-ésimo* autovetor calculado e F_{vi} é a *i-ésima* matriz diagonal de ponderação do *i-ésimo* autovetor. O símbolo $*$ representa o transposto do complexo conjugado do termo à sua esquerda.

As duas funções de custo definidas nos parágrafos anteriores e os critérios prévios de seleção, que são: as restrições de sensibilidade dos autovalores (Rs_i), a relação (2.106), as faixas de restrição dos autovalores ($R\lambda_i$), a relação (2.107), e o critério de otimalidade de *Pareto* (*COP*) formam combinados as estruturas do time de *fitness*. Outro critério de restrição que tem por base a sensibilidade dos autovalores, chamado de maior sensibilidade s_i de um indivíduo ($s_{max}I$), é conflitante com o *COP* e atua da seguinte maneira: o novo indivíduo só será candidato a ser incorporado à população permanente se cada uma das sensibilidades s_i for menor do que a maior sensibilidade de qualquer indivíduo da população permanente. Os conjuntos das funções de custo e dos critérios prévios são:

$$F_{custo} = \{fc_i|(J_S, J_E), \qquad i = 1, \ldots, n_{fc}\} \tag{4.25}$$

onde F_{custo} representa o conjunto de funções de custo, n_{fc} é a quantidade de funcões de custo do time de *fitness*, no caso duas, e

$$C_{PSelec} = \{cps_j | Rs_i, R\lambda_i, COP, s_{max}I\}, \quad i = 1, \ldots, n; \quad j = 1, \ldots, n_{pselec} \tag{4.26}$$

representa o conjunto de critérios prévios de seleção, n representa a ordem do sistema dinâmico, equação (2.97) e n_{pselec} é a quantidade de critérios prévios, no caso quatro.

O time de funções de *fitness* constituído por cinco estruturas de funções é representado pelo seguinte conjunto:

$$TF^5_{fitness} = \{ff_1, ff_2, ff_3, ff_4, ff_5\} \tag{4.27}$$

onde cada estrutura de *fitness* é constituída por uma função de custo e um conjunto de critérios prévios de seleção:

$$ff_1 = \{J_S, (Rs_i, R\lambda_i, COP), \quad i = 1, \ldots, n\} \tag{4.28}$$
$$ff_2 = \{J_S, (Rs_i, R\lambda_i, s_{max}I), \quad i = 1, \ldots, n\} \tag{4.29}$$
$$ff_3 = \{J_E, (Rs_i, R\lambda_i, COP), \quad i = 1, \ldots, n\} \tag{4.30}$$
$$ff_4 = \{J_E, (Rs_i, R\lambda_i, s_{max}I), \quad i = 1, \ldots, n\} \tag{4.31}$$
$$ff_5 = \{J_E, R\lambda_i, \quad i = 1, \ldots, n\} \tag{4.32}$$

Capítulo 5

A Plataforma Paralela APAE-AGMO/RLQ

5.1 Introdução

Os algoritmos do *time de funções de fitness* e da unidade de decisão são as principais contribuições apresentadas neste capítulo. O *time de funções de fitness* representa os mais variados ambientes no intuito de capturar a maior quantidade de indivíduos-QR que satisfaçam as especificações de projeto, isto é, a grosso modo pode-se afirmar que os vários ambientes são diferentes formas de expressar a mesma situação, pois todos eles são construídos a partir das mesmas especificações de projeto. A unidade de decisão tem por objetivo direcionar a busca, intensificar a busca em determinadas regiões e promover a diversidade genética de indivíduos-QR.

As técnicas de otimização multiobjetivo para formular o problema de controle, capítulo (2), e de computação evolutiva para solucionar o problema, capítulo (3), bem como os mecanismos para incrementar a velocidade e a qualidade da solução, capítulo (4), são os três elementos que foram utilizados no desenvolvimento de um arcabouço para resolver o problema de alocação de autoestruturas. Especificamente, este arcabouço de solução é constituído do problema de alocação de autoestruturas formulado como um problema de otimização multiobjetivo tendo por base o projeto do regulador linear quadrático e o método das desigualdades, um algoritmo genético (AG) para realizar a busca de matrizes de ponderação do projeto (RLQ) e uma unidade lógica de decisão (ULD) para guiar a busca do AG.

A estruturação deste arcabouço como uma ferramenta para resolver o problema de alocação de autoestruturas (*AAE*) está fundamentada nos três elementos citados no parágrafo anterior e a sua união, como uma única entidade, é construída em cima de técnicas de programação paralela. Os elementos da paralelização são o paradigma de programação paralela programa-único e dados-múltiplos (*SPMD*) e uma biblioteca de programação paralela *Message-Passing*. Este arcabouço é chamado de plataforma para alocação paralela de autoestruturas via algoritmo genético multiobjetivo e projeto *RLQ*, cujo acrônimo é *APAE-AGMO/RLQ*.

A paralelização da solução do problema *AAE* é realizada em dois níveis. O primeiro nível trata da paralelização do algoritmo genético e o segundo nível trata da solução simultânea de diversos projetos *RLQ*.

O diagrama de blocos, Figura (5.1), apresenta uma visão geral do algoritmo paralelo desenvolvido para alocar autoestruturas de sistemas dinâmicos. A Figura mostra que a plataforma *APAE-AGMO/RLQ* é formada por um conjunto de algoritmos genéticos multiobjetivo (*AGMO's*) e que por sua vez cada *AGMO* é constituído de um otimizador genético e de uma unidade de decisão.

Os eventos na plataforma *APAE-AGMO/RLQ*, Figura (5.1), ocorrem nas formas sequencial e paralela, considerando o sincronismo no início do *ciclo de busca*. Na primeira forma sequencial, o coordenador, módulo $AGMO_C$ recebe os dados de entrada (o modelo do sistema dinâmico em variáveis de estado, as restrições do problema e os parâmetros de simulação), gera uma população inicial de indivíduos-QR e distribui esta população entre os $AGMO's$ coordenados. Na forma paralela, todos os $AGMO's$ iniciam as iterações do *ciclo de busca* e quando algum dos critérios de parada é satisfeito a população permanente é enviada para o *AGMO* coordenador. Na segunda forma sequencial, o $AGMO_C$ recebe as populações finais de cada *AGMO* coordenado, escolhe os melhores indivíduos e pode realizar uma busca final. Caso não se considere o sincronismo para iniciar o *ciclo de busca*, a ocorrência dos seguintes eventos na forma sequencial deve ser considerada: logo após o recebimento da população inicial pelo primeiro *AGMO* coordenado, ocorrem as primeiras iterações do *ciclo de busca* e só após o recebimento da população inicial por todos os processadores ocorre a fase paralela; pode existir um caso em que não ocorra uma busca simultânea por todos os $AGMO's$, ou seja, um dos *AGMO* convergiu antes do recebimento da população inicial pelo último processador.

O algoritmo que concretiza a plataforma *APAE-AGMO/RLQ*, os ambien-

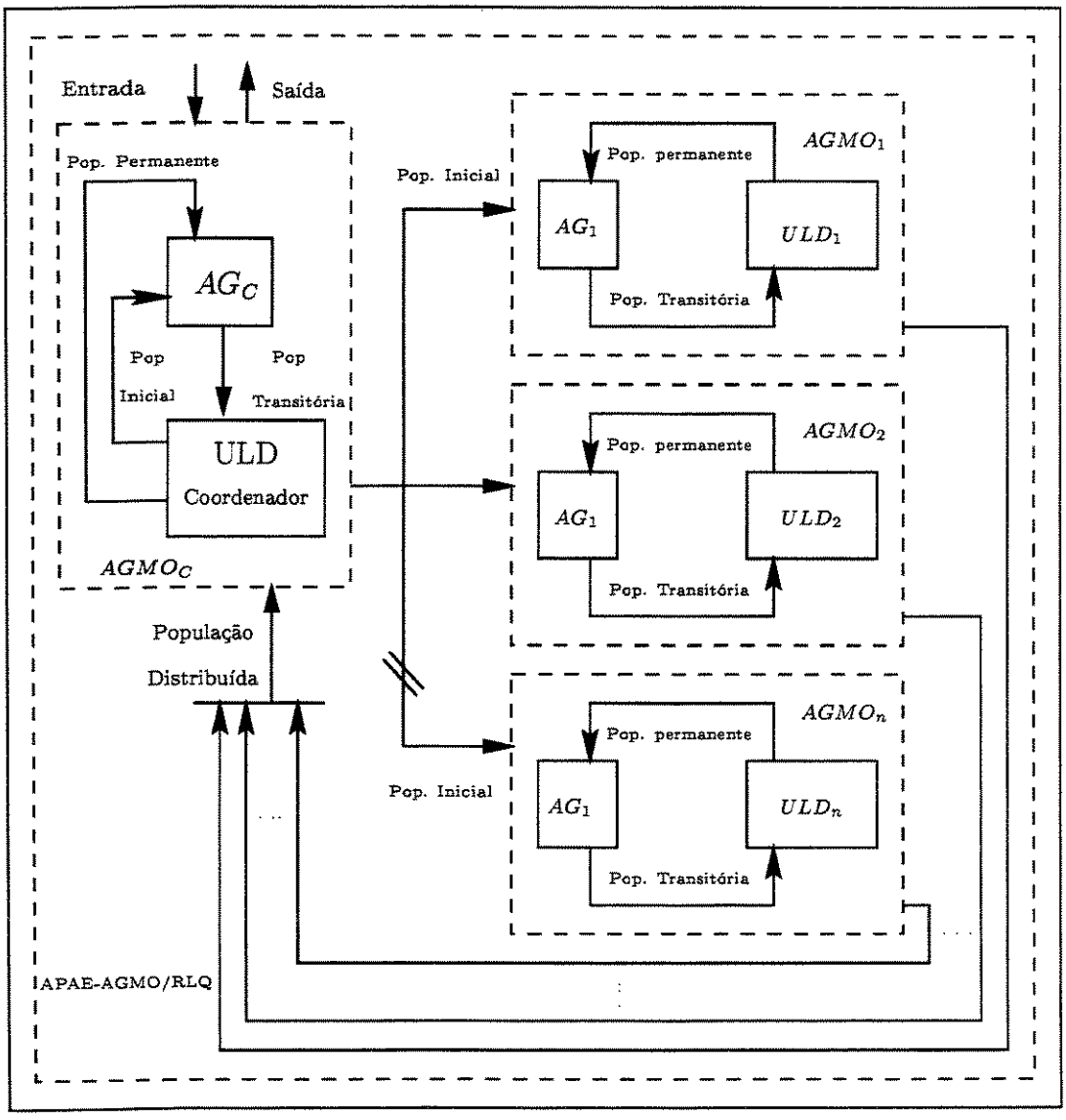

Figura 5.1: Interações entre *AGMO's*.

tes de paralelização, o paradigma de programação paralela, as bibliotecas para paralelização, os algoritmos para o time de estruturas de funções de *fitness* e a unidade de decisão são os assuntos abordados nas seções deste capítulo. Os parágrafos a seguir descrevem o conteúdo de cada seção.

A seção (5.2) descreve os passos que formam o algoritmo paralelo genético multiobjetivo para a alocação de autoestruturas. Inicialmente, apresenta-se o algoritmo geral e descreve-se a funcionalidade de cada declaração. Os algoritmos do otimizador genético, subseção (5.2.1), da unidade de decisão, subseção (5.2.2), e das estruturas de *fitness*, subseção (5.2.3) são os procedimentos básicos que formam o algoritmo proposto, e para cada declaração destes procedimentos apresentam-se a funcionalidade, as entradas e as saídas.

A seção (5.3) apresenta a justificativa da classificação dos ambientes de paralelização, que são constituídos de redes de estação de trabalho. Aspectos sobre a implementação do algoritmo Paralelo *AAE*-Genético, tais como: o paradigma de programação paralela, as bibliotecas de paralelização e as características dos sistemas computacionais, são abordados na seção (5.4).

5.2 O Algoritmo Paralelo AAE-Genético

As características do algoritmo paralelo para alocação de autoestruturas, a unidade de decisão, paradigma de programação paralela, linguagem de programação paralela e as formas de comunicação são os tópicos abordados nesta seção.

A seguir, apresenta-se a estrutura geral do algoritmo proposto e define-se a funcionalidade de cada declaração ou passo. O algoritmo é formado por dois módulos sequenciais e um paralelo. Os passos do algoritmo são:

\ Parte Sequencial 1 - Inicializações
Inicializar Variáveis
SE Coordenador
  Entrada ← [dados do sistemas dinâmico,
    parâmetros de simulação]
  $Pop_{inicial}$ ← [$Ger_{pop}(QR)$,semente]
  $Apop_{inicial}$ ← [$Ppop_{inicial}$, rank(QR)]
  $Opop_{inicial}$ ← [$Ppop_{inicial}$, rank(QR),fft]
FIM SE Coordenador
\ Parte Sequencial 1 - Modo de Comunicação Mestre/Escravo

SE Coordenador
$Enviar_{coordenador} \leftarrow [Entrada, Opop_{inicial}]$
SE NÃO
$Receber_{coordenados} \leftarrow [Entrada, Opop_{inicial}]$
FIM SE Coordenador
\ Parte Paralela - Ciclo de Busca
$Pop_{QR} \leftarrow Opop_{inicial}$
INICIO Ciclo de Busca
$Ciclo_{busca} \leftarrow [Pop_{QR}, Oper_{QR}, E_{busca}, Crit_p, Apop_{QR}, Opop_{QR}, Ipop_{QR}]$
FIM Ciclo de Busca
$Pop_{permanente} \leftarrow Pop_{QR}$
\ Parte Sequencial 2 - Modo de Comunicação Escravo/Mestre
SE Coordenador
$Receber_{coordenador} \leftarrow [Pop_{permanente}, Busca_{perf}]$
SE NÃO
$Envia_{coordenado} \leftarrow [Pop_{permanente}, Busca_{perf}]$
FIM SE Coordenador
$Saida_{QR} \leftarrow [Pop_{permanente}, Autoestrutura, Controladores, Busca_{perf}]$

A descrição da funcionalidade de cada declaração e seus parâmetros, as entradas e os retornos de cada função, são apresentadas a seguir:

- Parte Sequencial 1 - Inicializações
  - A função $Entrada[]$ retorna os dados do sistemas dinâmico, a autoestrutura especificada e os parâmetros de simulação fixados pelo projetista.
  - A função $Pop_{inicial}[]$ tem como entrada uma semente para o gerador aleatório e retorna a população inicial, $Ger_{pop}(QR)$, com uma quantidade de n_{indiv} indivíduos-QR.
  - A função $Apop_{inicial}[]$ tem como entrada a população inicial, $Ppop_{inicial}$, e retorna a sua pontuação, $rank(QR)$.
  - A função $Opop_{inicial}[]$ retorna a população inicial ordenada e tem como entrada a população inicial pontuada, $Ppop_{inicial}$, e o tipo da função de custo, *fft*, que está sendo considerada na ordenação.

- Parte Sequencial 1 - Modo de Comunicação Mestre/Escravo
  - A função $Enviar_{coordenador}[]$ envia os dados de entrada e a população inicial para os processos coordenados.
  - A função $Receber_{coordenados}[]$ recebe os dados de entrada e a população inicial ordenada enviada pelo processo coordenador.
- Parte Paralela - Ciclo de Busca
  - A função $Ciclo_{busca}[]$ retorna uma nova população permanente Pop^{t}_{QR} e tem como entrada a população permanente, Pop^{t-1}_{QR}, na iteração $t-1$ do *ciclo de busca*, as operações cromossômicas, $Oper_{QR}$, estratégia estabelecida de busca, E_{busca}, critérios de parada, $Crit_{p}[]$ e as funções que realizam a avaliação, $Apop_{QR}[]$ dos novos indivíduos-QR, inclusão de novos indivíduos, $Ipop_{QR}[]$, na população permanente e a ordenação da nova população.
- Parte Sequencial 2 - Modo de Comunicação Escravo/Mestre
  - A função $Receber_{coordenador}[]$ recebe a população permanente, $Pop_{permanente}$, de cada processo coordenado e o perfil do desempenho de cada busca, $Busca_{perf}$.
  - A função $Envia_{coordenado}[]$ envia a população permanente de cada processo coordenado e o perfil do desempenho de cada busca, $Busca_{perf}$.
- Parte Sequencial 2 - Relatório de Saída
  - A função $Saida_{QR}[]$ cria um relatório das autoestruturas, dos controladores e do perfil de desempenho da busca.

5.2.1 O Algoritmo do Otimizador Genético

O algoritmo do otimizador genético (otimizador-AG) é essencialmente um mecanismo de busca da parte paralela do algoritmo AAE-Genético apresentado na seção (5.2). As principais declarações do algoritmo otimizador-AG e a sua funcionalidade são discutidas nos parágrafos subsequentes. Os seguintes passos formam o *ciclo de busca*:

INICIO *Ciclo de Busca*
$Oper_{QR} \leftarrow [Pop_{QR}^{t-1}, T_{oper}, E_{ud}]$
$Apop_{QR} \leftarrow [Pop_{QR}^{t}, rank(QR), T_{fitness}]$
$Opop_{QR} \leftarrow [APop_{QR}^{t}, rank(QR)]$
$Crit_{p} \leftarrow [N_{i}ter, C_{conv}]$
$U_{dec} \leftarrow [Tpop_{QR}, E_{ud}]$
FIM *Ciclo de Busca*

Agora, descreve-se a funcionalidade de cada declaração:

- A função $Oper_{QR}[]$ retorna uma população-QR, Pop_{QR}^{t}, que é montada a partir de indivíduos da população, Pop_{QR}^{t-1}, da iteração anterior, $t-1$, e de indivíduos provenientes de alguma operação cromossômica, T_{oper}, definida através de um sorteio e cuja busca pode ter seguido alguma estratégia definida em E_{ud}.

- A função $Apop_{QR}[]$ retorna a população, Pop_{QR}^{t}, pontuada, $rank(QR)$, segundo os critérios estabelecidos por alguma estrutura de função de *fitness* e a variável T_{fft} define qual é a estrutura escolhida pelo projetista.

- A função $Opop_{QR}[]$ ordena a população-QR pontuada, $APop_{QR}^{t}$.

- A função $Crit_{p}[]$ verifica se a população da iteração t satisfaz um dos dois critérios de parada. O primeiro critério, verifica se as iterações do *ciclo de busca* atingiram o limite , N_{iter}. O segundo critério, verifica se a cada iteração a autoestrutura desejada foi atingida, C_{conv}.

- A função $U_{dec}[]$ retorna uma das estratégias de busca, E_{ud}, previamente estabelecida tendo por base a população transitória $Tpop_{QR}$ e condicionada à ocorrência da operação *crossover*.

O algoritmo do otimizador genético é constituído de 5 passos, sendo que na sua essência quatro passos definem o otimizador, porque o último passo descreve a unidade de decisão e esta não é considerada como parte essencial do processo de busca, já que o otimizador pode efetuar o processo de busca independentemente das estratégias estabelecidas nesta unidade.

5.2.2 O Algoritmo da Unidade de Decisão

A operação *crossover* é o ponto do otimizador-*GA* em que as estratégias previamente estabelecidas na unidade de decisão são implementadas no sentido de direcionar a busca, de evitar a perda de diversidade genética dos indivíduos e de intensificar a busca em determinadas regiões do espaço de solução. As estratégias para direcionar são o controle de parâmetros da operação *crossover*, a articulação de preferências, o critério da otimalidade de *Pareto*, o *fitness* médio da função de custo e as melhorias da função de custo. As estratégias que evitam a perda de diversidade têm por base o teorema do *schemata* e o paradigma do *multiarmed bandit*. As estratégias que visam uma exploração intensiva de regiões do espaço de busca têm por base indução de nichos artificiais e naturais, restrição de acasalamento (*mating*) e pequenas variações no grau de combinação entre dois indivíduos que participam da operação de *crossover*.

O algoritmo da unidade de decisão é constituído de três declarações básicas; cada uma representa uma das categorias de estratégia. A seguir, o algoritmo da unidade de decisão é apresentado:

INICIO *Unidade de Decisão*
$UD_{sent} \leftarrow [TPop^{t}_{QR},\ T_{efft},\ E_{sent},\ T_{oper}\ ]$
$UD_{diver} \leftarrow [TPop^{t}_{QR},\ E_{diver}]$
$UD_{inten} \leftarrow [TPop^{t}_{QR},\ T_{efft},\ E_{inten}]$
$E_{ud} \leftarrow (E_{sent}\ , E_{diver}\ ,\ E_{inten})$
FIM *Unidade de Decisão*

onde

- A função $UD_{sent}[]$ retorna um valor que estabelece se alguma estratégia de direcionamento, E_{sent}, deve ser implementada ou não. A decisão sobre a sua implementação tem por base a população transitória, $TPop^{t}_{QR}$, o tipo de estrutura de função de *fitness* e o tipo de operação cromossômica, T_{oper}.

- A função $UD_{diver}[]$ retorna um valor que estabelece se alguma estratégia de diversificação, E_{diver}, deve ser implementada ou não. A decisão sobre a sua implementação tem por base a semelhança entre os indivíduos-QR da população transitória.

- A função $UD_{inten}[]$ retorna um valor, E_{inten}, que estabelece se alguma estratégia de intensificação deve ser implementada ou não. A decisão sobre a sua implementação tem por base a população transitória, estratégias definidas pelo projetista e certos padrões definidos por características comuns de certos indivíduos-QR.

- O vetor $E_{ud}[]$ armazena as decisões, E_{sent}, E_{diver} e E_{inten}, resultantes das três categorias de estratégias definidas na unidade de decisão.

As três funções da unidade de decisão podem atuar de forma conjunta; a sua forma de atuação depende exclusivamente das medidas fornecidas pelas estruturas de *fitness*, do tipo de operação cromossômica e das estratégias definidas pelo projetista.

5.2.3 O Algoritmo do Time de Fitness

O algoritmo do *time de fitness* possui duas atribuições que são: pontuar os indivíduos no final de cada iteração do *ciclo de busca* e fornecer informações à unidade de decisão a fim de que esta dispare a estratégia para ajuste de parâmetros do otimizador-AG. A grosso modo, diz-se que as estruturas de *fitness* têm por objetivo realizar cálculos, verificar restrições e ordenar indivíduos.

O conjunto solução inicial ou população inicial corresponde a um grupo de pontos gerados de forma aleatória e cada ponto é chamado de indivíduo. Cada indivíduo da população é bem definido por um par de matrizes de ponderação e suas características podem ser observadas através de um vetor de sensibilidades normalizadas dos autovalores, de um conjunto de autovalores, de uma função de custo e dos ganhos do controlador. Então, cada indivíduo, chamado de $indiv_{QR}$, é representado por:

$$indiv_{QR} = \{crom_{QR}, (\Delta, \Lambda, J, K)\} \tag{5.1}$$

onde $crom_{QR}$ representa as matrizes de ponderação codificadas como um cromossomo artificial, Δ representa as sensibilidades normalizadas, Λ representa os autovalores, J representa a função de custo e K representa os ganhos do controlador.

A equação (2.104) fornece os elementos do vetor das sensibilidades e sua normalização é obtida quando divide-se a *i-ésima* sensibilidade S_i pela *i-*

ésima especificação de projeto ϵ_i. Então, o vetor das sensibilidades normalizadas é dado por:

$$\Delta^k = \{s_1, s_2, \ldots, s_n\} \tag{5.2}$$

onde s_i, $i = 1, \ldots, n$, representa as n sensibilidades dos autovalores do sistema de malha fechada na *k-ésima* iteração do *ciclo de busca*, equação (2.102), e n é a ordem deste sistema.

O algoritmo de *fitness* representando as cinco estruturas do *time de funções de fitness* é apresentado como um único bloco. Esta representação tem por base o modelo de programação paralela utilizado no desenvolvimento desta tese que é o paradigma de programação paralela *programa-único e múltiplos-dados*, cujo acrônimo na língua inglesa é *SPMD*.

A Tabela (5.1) apresenta, de forma macroscópica, os elementos que compõem cada estrutura do *time de funções de fitness* e o algoritmo apresenta em detalhes a constituição de cada um destes elementos e sua forma de atuação. Nesta tabela, as abreviações ff_i, $i = 1, \ldots, 5$, representam as estruturas de *fitness*, J_S representa a função de custo soma das sensibilidades, J_E representa a função de custo erro quadrático da autoestrutura, S_iR representa as restrições das sensibilidades, λ_iR representa as restrições dos autovalores, COP representa a atuação do critério da otimalidade de *Pareto* e IGS_i representa a restrição da pior sensibilidade.

	ff_1	ff_2	ff_3	ff_4	ff_5
Função de Custo	J_S	J_S	J_E	J_E	J_E
Critério prévio de Seleção	s_iR	s_iR	s_iR	s_iR	
	λ_{ci}R	λ_{ci}R	λ_{ci}R	λ_{ci}R	λ_{ci}R
	COP	IGs_iC	COP	IGs_iC	

Tabela 5.1: Estrutura Básica do *time de funções de fitness*

Duas considerações devem ser observadas na descrição dos passos do algoritmo de *fitness*. Supõe-se avaliação de um único indivíduo-QR e que o melhor indivíduo da iteração $k-1$ está representado pelo conjunto:

$$indiv_{QR}^{atual} = \{crom_{QR}^{atual}, (\Delta^{atual}, \Lambda^{atual}, J^{atual}, K^{atual})\} \tag{5.3}$$

O algoritmo descrito representa o k-ésimo passo de iteração do *ciclo de busca*. Então:

- **INÍCIO** Estrutura de *Fitness*

1. Cálculo do vetor das funções de desempenho normalizadas:
   - **PARA** ff_j , $j = 1, ..., 4$ **FAÇA**

$$\Delta^k = (s_1(K^k), s_2(K^k), ..., s_n(K^k)) \tag{5.4}$$

$$\text{onde } s_i(K^k) = \frac{S_i(K^k)}{\epsilon_i}, \quad i = 1, 2, ..., n$$

   - **FIM PARA**
2. Cálculo das funções de custo:
   (a) Soma das funções desempenho normalizadas:
   - **PARA** ff_j , $j = 1$ e 2 **FAÇA**

$$J_S^k = \sum_{i=1}^{n} s_i(K^k), \quad i = 1, 2, ..., n \tag{5.5}$$

   - **FIM PARA**

   (b) Função de custo erro quadrático:
   - **PARA** ff_j , $j = 3, ..., 5$ **FAÇA**

$$J_E^k = J_E(K^k) \tag{5.6}$$

   - **FIM PARA**
3. Ordenamento das restrições:
   - Todos os critérios de ordenamento consideram os limites dos autovalores:

$$\lambda_{ei} \leq \lambda_{ci}(K^k) \leq \lambda_{di}, \quad i = 1, ..., n \tag{5.7}$$

   (a) Ordenamento do maior $s_i(K^k)$:
   - **PARA** ff_j , $j = 2$ e 4 **FAÇA**

$$(a1) \quad \max_{K^k}\{s_i(K^k)\} < \max\{s_i\}^{atual}, \quad i = 1, 2, ..., n \tag{5.8}$$

   - **FIM PARA**

   ou

– **PARA** ff_j , $j = 2$ **FAÇA**

$$(a2)\ \max_{K^k}\{\Delta^k\} = \max\{\Delta_{atual}\} \quad e \quad J_S^k < J_S^{atual} \tag{5.9}$$

– **FIM PARA**

– **PARA** ff_j , $j = 4$ **FAÇA**

$$(a3)\ \max\{\Delta^k\} = \max\{\Delta_{atual}\} \quad e \quad J_E^k < J_E^{atual} \tag{5.10}$$

– **FIM PARA**

(b) $s_i(K^k)$ Ordenamento COP:

– **PARA** ff_j , $j = 1$ e 3 **FAÇA**

$$(b1)\ s_i(K^k)\} < \Delta_{atual}^i, \quad i = 1, 2, ..., n \tag{5.11}$$

– **FIM PARA**

ou

– **PARA** ff_j , $j = 1$ **FAÇA**

$$(b2)\ \Delta^k \leq \Delta_{atual}^i \quad e \quad J_S^k < J_S^{atual}, \quad i = 1, 2, ..., n \tag{5.12}$$

– **FIM PARA**

ou

– **PARA** ff_j , $j = 3$ **FAÇA**

$$(b3)\ \Delta^k \leq \Delta_{atual}^i \quad e \quad J_E^k < J_E^{atual}, \quad i = 1, 2, ..., n \tag{5.13}$$

– **FIM PARA**

(c) Ordenamento pelos limites dos autovalores:

– **PARA** ff_j , $j = 5$ **FAÇA**

$$J_E^k < J_E^{atual} \tag{5.14}$$

– **FIM PARA**

- **FIM** Estrutura de *Fitness*

5.3 Os Ambientes de Paralelização

A classificação dos ambientes para paralelização utilizados nesta tese tem por base a classificação proposta por (Bertsekas e Tsitsiklis, 1989). Esta classifica os ambientes paralelos como sendo aqueles que possuem as seguintes características: os processadores dedicam-se exclusivamente à solução do problema em questão, os processadores estão fisicamente próximos e a comunicação entre processadores é confiável e previsível. Quanto a sistemas distribuídos, estes são classificados como sendo aqueles em que os processadores não estão dedicados exclusivamente a solucionar o problema em questão, podem não estar fisicamente próximos e a comunicação pode não ser confiável no nível de *links* e imprevisível devido a atrasos.

Nesta tese, o ambiente constituído por computadores de alto desempenho, que podem dedicar-se exclusivamente à execução de um único programa do usuário, é denominado de ambiente exclusivamente paralelo (*AEP*) e o ambiente constituído por ambientes distribuídos é denominado de ambiente distribuído paralelo (*ADP*).

O ambiente distribuído paralelo é constituído de uma rede local de estações de trabalho que pode ser vista como uma arquitetura de processamento paralelo *MIMD* construída a partir de multi-computador. A inserção da rede de estações de trabalho nesta classificação tem por base a taxonomia de arquiteturas paralelas apresentada em (Zargham, 1995).

O algoritmo foi implantado em dois ambientes computacionais. O primeiro é uma rede de estações de trabalho que pode ser considerada um ambiente virtualmente paralelo ou ambiente distribuído. O segundo é um ambiente essencialmente paralelo, pois os processadores podem dedicar-se exclusivamente à execução de uma tarefa específica por vez. A subseção (5.4.2) descreve as topologias e as características destes dois ambientes computacionais.

5.4 A Implementação da Plataforma

De acordo com a concepção do algoritmo proposto, o paradigma de programação paralela é considerado: a) trivialmente paralelo, porque as tarefas dos processos de busca operam de forma independente, b) troca de mensagens, devido a característica de memórias distribuídas dos ambientes computacionais utilizados na implementação, e c) *SPMD*, porque todos os processadores executam o mesmo código e processam diferentes tipos de dados.

Esta seção descreve as bibliotecas e os sistemas computacionais utilizados para implementar o algoritmo paralelo, chamado de plataforma paralela, para alocação de autoestruturas de sistemas dinâmicos.

5.4.1 As Bibliotecas de Programação Paralela

As bibliotecas de programação paralela utilizadas na implementação do algoritmo proposto têm por base o modelo *message passing*. A principal característica do modelo *message passing* é que os processos comunicam-se entre si pelo envio de mensagens, tendo por base transmissão de dados (*send-receive*) via uma rede de interconexão; isto significa que não existe o conceito de memória compartilhada ou de processadores acessando diretamente a memória de outros processadores, (Donald *et al.*, 1998). A referência (Gropp *et al.*, 1994) caracteriza este modelo como sendo um conjunto de processos que possuem somente memória local e que comunicam-se entre si pelo envio e recebimento de mensagens.

As bibliotecas para programação paralela **MPI**, *Message Passing Interface*, e **PVM**, *Parallel Virtual Machine* foram as duas bibliotecas de programação utilizadas para implementar o algoritmo. Uma diferença marcante entre as duas bibliotecas é que a biblioteca **MPI** é exclusivamente projetada para implementar algoritmos *SPMD*, enquanto que a biblioteca **PVM**, além de aceitar este paradigma permite a implementação de algoritmos no paradigma *mestre-escravo*. Outra diferença entre as bibliotecas é que o **MPI** é padronizado e o **PVM** não possui qualquer compromisso com padronização e manutenção. Um ponto comum entre as duas bibliotecas é que ambas permitem que um conjunto de computadores seja visto como uma única máquina paralela virtual. O Apêndice (C) apresenta uma breve descrição destas duas bibliotecas e das primitivas utilizadas na implementação do algoritmo proposto nesta tese.

A biblioteca **PVM** é formada por dois elementos básicos. O primeiro é um programa processado por todas as máquinas que compõem a máquina virtual e que permite que o usuário processe uma aplicação **PVM** a partir de qualquer máquina; este programa é chamado de *daemon*. O segundo elemento é a biblioteca **PVM** que contém rotinas para distribuir processos, passagem de mensagens, coordenação de tarefas e modificação da máquina virtual, (Geist *et al.*, 1994).

A biblioteca **MPI** é o resultado da padronização de uma biblioteca de rotinas que são utilizadas para efetivar a comunicação entre computadores mas-

sivamente paralelos e *clusters* de estações de trabalho e redes heterogêneas, (Gropp *et al.*, 1994). Uma vantagem do **MPI** sobre o **PVM** é que um conjunto de seis rotinas é suficiente para estabelecer a comunicação entre processos.

As primitivas das bibliotecas **PVM** e **MPI** utilizadas nesta tese podem ser encontradas na referências (Geist *et al.*, 1994) e (Gropp *et al.*, 1994), respectivamente.

5.4.2 Os Sistemas Computacionais

Conforme estabelecido na seção (5.3), dois ambientes computacionais foram definidos com o intuito de verificar a eficácia do algoritmo proposto. O primeiro ambiente é formado por uma rede homogênea de estações de trabalho da *SUN*, sendo identificada como o *ADP* (ambiente distribuído paralelo). O segundo ambiente é uma máquina paralela *IBM 9076 SP2* que é um computador de memória distribuída *MIMD*, sendo identificado como o *AEP* (ambiente exclusivamente paralelo). As características de cada ambiente são apresentadas no Apêndice (B).

Capítulo 6

Resultados

6.1 Introdução

O desempenho do algoritmo proposto é verificado sob dois aspectos. O primeiro aspecto enfoca a habilidade do algoritmo em alocar autoestruturas de sistemas dinâmicos. O segundo aspecto enfoca o desempenho do algoritmo em relação aos mecanismos de busca, tais como: o otimizador-AG e a unidade de decisão.

Os resultados analisados são provenientes de uma rede de estações de trabalho e de um ambiente computacional de alto desempenho, cujas características estão descritas no Apêndice (B). As bibliotecas de programação paralela utilizadas na implementação foram **PVM** e **MPI**.

Os resultados, não só, apresentam o desempenho do método proposto para alocação de autoestruturas, como também, traduzem pela sequência de apresentação, as etapas do desenvolvimento do algoritmo. Inicialmente são apresentados resultados obtidos sem a unidade de decisão, salientando-se a evolução da alocação da autoestrutura para iterações do *ciclo de busca*. O segundo resultado apresenta o desempenho do algoritmo genético com estratégias, tendo por base a teoria do *schemata*, paradigma do *multiarmed bandit* e o time de *fitness*, que são implementadas na unidade de decisão. O terceiro tipo de resultado apresenta o desempenho do algoritmo quando implementa-se regras interativas, na unidade de decisão, que ajustam os parâmetros da operação *crossover* após o término do processo de busca. O quarto tipo de resultado apresenta os desempenhos dos controladores e do otimizador quando as regras para o ajuste dos parâmetros da operação cro-

mossômica *X-over* são implementadas *on-line*. O último tipo de resultado dedica-se a análise dos efeitos da alocação da autoestrutura, imposta pelos ganhos do controlador por realimentação de estado, enfatizando os acoplamentos entrada-estado e estado-saída.

6.2 O Sistema Dinâmico Teste

Para teste e ressaltar as qualidades da metodologia proposta nesta tese escolheu-se para sistema teste o modelo de uma aeronave da *Lockheed, tipo L1011 Tristar*. Este sistema foi utilizado por (Davis e Clarke, 1995) para realizar uma alocação de autoestruturas, alocar os autovalores em uma determinada faixa e alocar os autovetores para o desacoplamento de modos. Tais autores utilizaram o projeto RLQ para determinar os ganhos do controlador por realimentação de estado e um algoritmo genético binário paralelo para determinar as matrizes de ponderação que devem satisfazer às especificações de projeto. O erro quadrático da diferença entre autoestruturas especificadas pelo projetista e obtidas durante o processo de busca, foi a função de *fitness* adotada para verificar se a autoestrutura obtida corresponde às especificações de projeto.

O modelo da aeronave é linearizado em uma condição de cruzeiro, (Sobel e Shapiro, 1985*a*). A sua representação em variáveis de estado é dada pelas equações (6.1) e (6.2):

$$\dot{x} = Ax + Bu \tag{6.1}$$

$$y = Cx \tag{6.2}$$

Com matriz de estado A :

$$A = \begin{bmatrix} -20.00 & 0.0000 & 0.0000 & 0.0000 & 0.000000 & 0.0000 \\ 0.0000 & -25.00 & 0.0000 & 0.0000 & 0.000000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.000000 & 0.0000 \\ -0.744 & -0.032 & 0.0000 & -1.540 & -0.00420 & 1.5400 \\ 0.3370 & -1.120 & 0.0000 & 0.2490 & -1.00000 & -5.200 \\ 0.0200 & 0.0000 & 0.0386 & -0.996 & -0.00029 & -0.117 \end{bmatrix} \tag{6.3}$$

vetor ponderação de entrada B:

$$B = \begin{bmatrix} 20.0 & 0.00 \\ 0.00 & 25.0 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.00 & 0.00 \\ 0.000 & 0.000 \end{bmatrix} \tag{6.4}$$

e matriz de saída C:

$$C = \begin{bmatrix} 0.0 & 0.0 & 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{6.5}$$

Os autovalores do sistema:

$$\begin{aligned} \lambda_1 &= -0.2276 & \lambda_2 &= -0.8955 \\ \lambda_{3,4} &= -0.7670 \pm 0.9251 & & \\ \lambda_5 &= -20.0 & \lambda_6 &= -25.0 \end{aligned}$$

6.3 Otimizador-AG sem UD

O primeiro grupo de resultados apresentados é proveniente de uma simulação computacional utilizando três máquinas do ambiente distribuído paralelo, bibliotecas de comunicação **PVM** e sem a implementação de estratégias da unidade de decisão. As estações utilizadas foram: *Águia - SPARCstation 20*, *Urano- SPARCstation 4* e *Saturno - SPARCstation 4*; a primeira é responsável pela distribuição dos processos *escravo* (coordenado) e realiza um processamento após o envio dos processos *escravo* para as máquinas *Saturno* e *Urano*. A população permanente é constituída de 10 indivíduos, isto é, a cada geração de uma nova população somente 10 indivíduos conseguem sobreviver às restrições impostas pelas função de *fitness*. Durante esta simulação foram geradas 1000 populações a partir de cada processo; isto totaliza um montante de 3000 populações quando contabilizam-se os três processos: *mestre* (coordenador) e os dois *escravos* (coordenados).

Os resultados apresentados a seguir estão divididos em duas partes. A primeira parte são comentários e análise da saída em forma de tabela. A segunda parte apresenta uma análise dos gráficos das funções de *fitness*, a quantidade de gerações, a autoestrutura e as sensibilidades dos autovalores.

6.3.1 Características da Simulação

Este item apresenta as características do algoritmo genético que podem ser definidas pelo projetista antes das simulações. Estas características envolvem a quantidade de indivíduos que são inseridos na população permanente após cada iteração do *ciclo de busca*, o tamanho da população permanente, a quantidade de indivíduos gerados a partir de cada operação cromossômica, o número de iterações do *ciclo de busca*, a magnitude dos elementos das matrizes de ponderação e a obtenção da semente do gerador aleatório.

A semente do gerador pseudo-aleatório é obtida a partir do relógio da máquina responsável pelo processo *mestre* e a sequência de números pseudo-aleatórios é utilizada para gerar a população inicial e para realizar as operações genéticas do processo, como também é utilizada em qualquer outro evento em que sua ocorrência dependa de aleatoriedade. As sementes dos processos *escravo* são geradas da semente do *mestre* através de um incremento para cada processo. Além do mais, a semente armazenada permite a reprodução dos resultados obtidos após o término da simulação.

A quantidade de indivíduos que serão inseridos nas novas populações depende da operação chamada seleção da nova população, definida no capítulo (3), seção (3.4.2), que varia de geração para geração de forma aleatória. Para esta simulação, a Tabela (6.1) mostra a quantidade de indivíduos que podem ser inseridos para os três processos (o *mestre* e os dois *escravos*). Ao fim de cada iteração do *ciclo de busca* os melhores indivíduos de cada população transitória e permanente são escolhidos para formar a população final. Em suma, esta opção estipula a quantidade máxima de indivíduos das novas gerações que são inseridos nas populações permanentes; com isto, espera-se reduzir a saturação das populações e ampliar a diversidade genética dos indivíduos que formam a população permanente.

Processo	*Probabilidade*			
	5%	5%	5%	85%
Mestre	1	5	2	1
Escravo-1	1	1	2	1
Escravo-2	1	1	1	1

Tabela 6.1: Probabilidade de inserção de novos indivíduos nas populações.

As probabilidades de ocorrência das operações cromossômicas são 10%,

50%, 20% e 10% para duplicação, *crossover*, mutação e visitante, respectivamente. Para o caso da mutação, onde se tem dois tipos, haverá um novo sorteio para escolher o tipo com 30% para a binária e 70% para decimal; se a mutação decimal for sorteada haverá um novo sorteio para definir se a operação será uma mutação global ou local com 50% de probabilidade para cada uma delas.

A faixa da representação binária de cada indivíduo-QR é caracterizada da seguinte maneira: cada elemento das matrizes Q ou R é visto como um gene e cada gene é representado por 16 dígitos binários, onde 8 dígitos representam a parte inteira e 8 representam a parte fracionária. Então, cada cromossomo-Q possui 336 dígitos binários e um indivíduo-QR possui 384 dígitos; este cálculo é realizado considerando que cada par de matrizes Q e R possui 24 elementos. Os valores mínimos e máximos dos elementos das matrizes são 0.0001 e 255, respectivamente.

O tamanho da população permanente e a quantidade de indivíduos gerados a partir das operações cromossômicas são estabelecidos pelo projetista. Para esta simulação, as seguintes quantidades foram estabelecidas: a população permanente é constituída de 10 indivíduos-QR, a operação *X-over* produz 14 indivíduos, as mutações produzem 5, a duplicação produz 2 e a operação visitante produz 1 indivíduo-QR, que juntos com a população permanente constituem a população transitória. Pode-se concluir que o tamanho da população transitória varia de acordo com o tipo de operação cromossômica ocorrido para cada iteração do *ciclo de busca*.

6.3.2 O Processamento Paralelo Distribuído

Este item enfatiza aspectos da utilização de recursos computacionais e o desempenho do otimizador-AG durante o processamento paralelo distribuído. As informações sobre a utilização do ambiente computacional são obtidas utilizando primitivas do **PVM**. Com relação ao otimizador-AG as informações apresentam a quantidade de cada tipo de operação genética realizada durante a busca, a maior sensibilidade do melhor indivíduo-QR de cada tarefa, o número de iterações do ciclo de busca e o valor atingido de tolerância.

A Tabela (6.2) mostra uma sinopse de algumas das informações obtidas a partir das funções da biblioteca **PVM**, associando a tarefa ou o processo com a máquina, o nome do processo, a identificação da tarefa e a arquitetura utilizada durante o processamento.

Os resultados do *Host-Águia* e das tarefas que foram distribuídas por este

Host, Tabela (6.3), estão organizados por ordem de chegada nesta máquina. O primeiro resultado apresentado é do *Host-Águia*, o segundo resultado vem do *Host-Saturno* e o terceiro resultado vem do *Host-Urano*. Na referida Tabela, são apresentados resultados globais, tais como: iteração em que a convergência foi atingida, quantidade de operações genéticas e tolerâncias (sensibilidades) atingidas em cada tarefa durante o *ciclo de busca*.

O tempo de simulação consumido pelos três processos foi de três minutos, 18 segundos e 48 centésimos de segundo (3:18:48) Este tempo foi obtido através do comando *timex* do sistema operacional *UNIX*.

O comportamento do otimizador-AG pode ser melhor observado e compreendido através de uma análise via gráficos, onde tem-se a oportunidade de verificar os comportamentos: da função de *fitness*, das sensibilidades dos autovalores e da frequência de ocorrência das operações genéticas para cada nova população. Agora apresenta-se uma série de resultados na forma gráfica.

A Figura (6.1) representa a pior sensibilidade do melhor indivíduo-QR após o término de cada iteração do *ciclo de busca* e é dada pela relação (3.21) do algoritmo de *fitness*, capítulo (3) , seção (3.4.2). Os gráficos mostram que houve uma convergência monotonicamente decrescente para um valor menor que 1 em todas as três tarefas em que o processo poderia ter sido interrompido sem maiores prejuízos em torno da população 780. Contudo, para as Tarefas 1 e 3 os respectivos *ciclos de busca* atingiram os seus melhores valores em torno da população 350, e para a Tarefa 1, exclusivamente, em torno da população 310. No tocante à satisfabilidade, as Tarefas 1, 2 e 3 poderiam ter sido interrompidas após as iterações 17, 735 e 212, respectivamente, porque nestas iterações os controladores já satisfazem todas as restrições impostas pelo projetista.

A função de custo, $J_S^k(QR)$, é dada pela relação (3.22) do algoritmo de *fitness*, capítulo (3), seção (3.4.2). Esta apresentam um comportamento com

HOST	Processo	Programa	Tarefa-ID	Arquitetura
Águia	1	*ga_mrq*	74379	SUNMP
Saturno	2	*ga_trab*	74601	SUNMP
Urano	3	*ga_trab*	74599	SUNMP

Tabela 6.2: Gerenciamento do Processamento Paralelo via **PVM**.

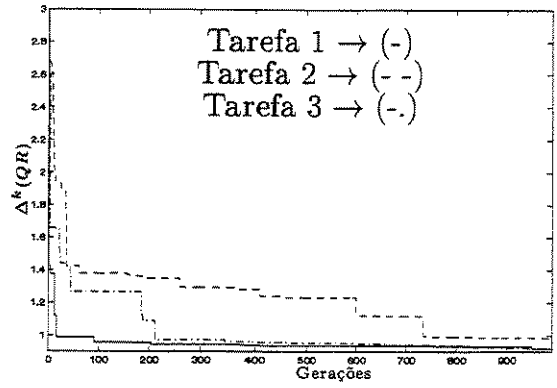

Figura 6.1: Evolução das piores sensibilidades *versus* Gerações para as três tarefas.

oscilações não periódicas para as Tarefas 2 e 3, Figura (6.2), sendo mais acentuada na Tarefa 2; isto pode ser explicado através da equação (3.17). Como pode-se verificar, equação (3.17), a função de custo é constituída da soma das sensibilidades e todas as sensibilidades podem estar acima ou abaixo dos seus novos valores durante a inserção do novo melhor indivíduo-QR na população permanente. Já no caso do processo 1 (*mestre*), observa-se que a função objetivo possui um comportamento monotonicamente descrescente e seus melhores progressos são obtidos nas primeiras 100 populações; em torno das populações 200 e 400 observa-se pequenas melhorias, que vem ocorrer em torno da população 900.

O Apêndice (D) apresenta o comportamento dos autovalores e das sensibilidades dos autovalores, considerando o melhor indivíduo de cada processo, durante iterações do *ciclo de busca*. As sensibilidades são utilizadas para

HOST	Processo	Semente	Tol.	Iterac.	Operações Genéticas			
					X-over	Mut.	Dup.	Visit.
Águia	1	38832393	0.98	17	558	210	76	116
Saturno	2	12944131	0.99	735	594	202	95	99
Urano	3	3236032	0.96	212	592	200	107	91

Tabela 6.3: Tarefas versus iterações e operações cromossômicas em cada ciclo.

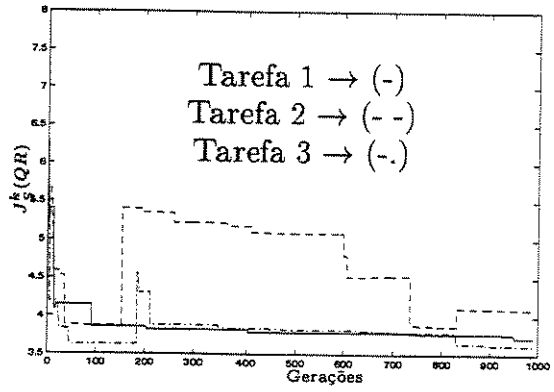

Figura 6.2: Função de Custo J_S *versus* gerações para as três tarefas.

explicar a natureza oscilatória da função de custo J_S.

Os ganhos dos controladores para as três tarefas estão apresentados na Tabela (6.4). Estes ganhos correspondem ao melhor indivíduo de cada tarefa. A seleção destes indivíduos teve como critérios: sensibilidades de autovalores e faixas de autovalores. A Tabela (6.5) apresenta as restrições referentes às sensibilidades dos autovalores que foram montadas através da implementação do controlador de referência, obtido em (Davis e Clarke, 1995), no modelo em variáveis de estado, equação (6.1); os limites das faixas de autovalores foram obtidos através de incrementos e decrementos dados aos respectivos autovalores.

Tabela 6.4: Ganhos dos Controladores.

Tarefa	Ganhos
1	0.449 0.005 -0.012 -0.745 -0.103 1.179
	0.007 0.135 -0.114 -0.509 -0.509 1.402
2	0.415 0.003 -0.022 -0.526 -0.068 0.741
	0.009 0.119 -0.197 -0.606 -0.552 1.582
3	0.446 0.004 0.013 -0.728 -0.088 1.127
	0.006 0.120 -0.054 -0.423 -0.433 1.190

O Apêndice (E) apresenta uma análise das medidas de tempo dos *ciclos de buscas* para oito tarefas. Salienta-se os benefícios advindos da instrumentação de tempo para implementações de algoritmos paralelos.

6.3.3 Conclusões e Comentários

Os resultados obtidos neste projeto foram satisfatórios sob o ponto de vista de alocar a autoestrutura especificada. Contudo, muitas tentativas foram efetuadas até o algoritmo atingir o nível de satisfabilidade requerido pelas especificações de projeto; pode-se dizer que o desempenho da busca em termos de velocidade de resposta foi satisfatório, apesar dos dois processos escravo consumirem uma boa quantidade do limite máximo de iterações do *ciclo de busca*, Tabela (6.3). A baixa velocidade de convergência e a quantidade de casos sem solução conduziram ao desenvolvimento de mecanismos para melhorar a qualidade da busca. Estes mecanismos são implementados na unidade lógica de decisão e os resultados de suas ações são apresentados nas próximas seções.

6.4 Otimizador-AG com *UD*

Os resultados obtidos com as estratégias implementadas na unidade de decisão (*UD*) para guiar a busca dos indivíduos-QR são apresentados nesta seção. As estratégias time de funções de *fitness*, seção (4.4), *schemata* e *MAB*, seção(4.3), são os três mecanismos desenvolvidos para melhorar a qualidade da busca que constituem a *UD*.

Analisa-se dois tipos de resultados. O primeiro tipo enfatiza o desempenho das estratégias implementadas na unidade de decisão e do otimizador-AG; estes resultados apresentam gráficos dos perfis das populações, gerações e indivíduos para cada estrutura do time de funções de *fitness*. O segundo tipo apresenta o desempenho do sistema dinâmico quando controladores são implementados no modelo em variáveis de estado, equações (6.1) e (6.2), e o sinal impulso unitário é aplicado na sua entrada.

6.4.1 As Simulações

O ambiente computacional utilizado é uma rede de estações de trabalho da *SUN*, cujas características estão apresentadas no Apêndice (B). Os resultados apresentados são provenientes de cinco tarefas; cada tarefa possui sua própria estrutura de *fitness*, definida conforme a Tabela (5.1), capítulo (5), subseção (5.2.3).

6.4.2 Os Desempenhos das Estratégias

Os desempenhos das estratégias são verificados através dos resultados fornecidos pelo otimizador-AG para 100 iterações do *ciclo de busca*. Os resultados são apresentados nas Figuras (6.3-6.5), e representam os comportamentos das funções de custo, da maior sensibilidade e do perfil da população final de cada tarefa.

As Figuras (6.3) e (6.4) são analisadas simultaneamente. Para a função de custo ff_1, observa-se que reduções em $s_i(K)$, Figura (6.4), provocam reduções na sua função de custo, Figura (6.4); isto acontece porque estas estruturas de *fitness* são dependentes do critério de otimalidade de *Pareto* (COP). Contudo este fenômeno não acontece para as funções de custo das outras estruturas. Os tempos gastos, típicos, nos *ciclos de busca* para as funcões de *fitness*, $ff_i, 1, \ldots, 4$, foram 62.33, 65.69, 119.50 e 66.64 segundos, respectivamente.

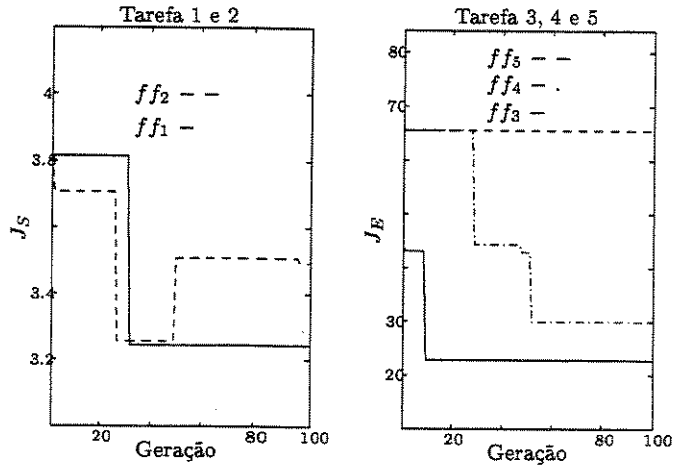

Figura 6.3: Gerações × comportamento da funções de custo para o melhor indivíduo.

A Figura (6.5) apresenta o perfil final da população, que é ordenada pelas maiores sensitividades de cada indivíduo que forma a população permanente. Observa-se que os melhores perfis foram obtidos para as estruturas ff_2 e ff_3.

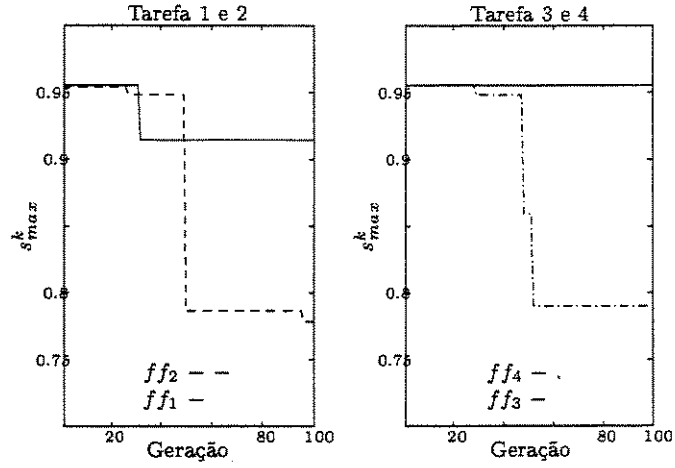

Figura 6.4: Geração × máximo s^k de ff_1, ff_2, ff_3, ff_4.

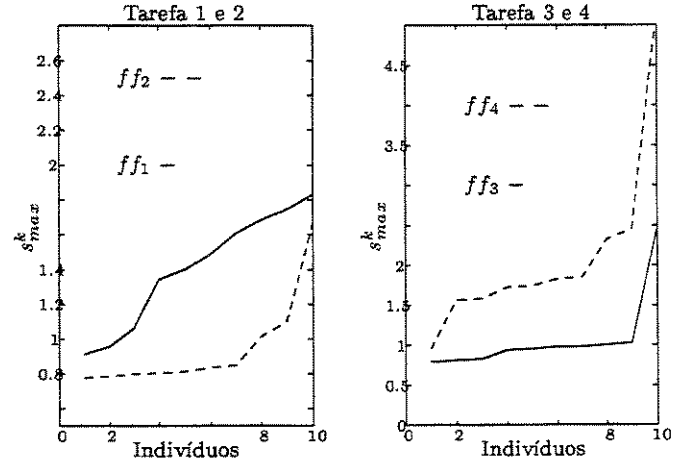

Figura 6.5: Perfil dos indivíduos da População final × máximo s^k para ff_1, ff_2, ff_3 e ff_4.

6.4.3 Os Desempenhos dos Controladores

Os desempenhos dos controladores são verificados através da comparação com o controlador de referência, (Davis e Clarke, 1995). As Figuras (6.6-6.8) apresentam as respostas ao impulso destes controladores, como pode ser visto estes apresentaram os melhores amortecimentos e para algumas variáveis de estado o tempo de acomodação foi maior. Contudo, como todos os cinco controladores foram obtidos de forma paralela, significando que são membros de um único processo, pode-se afirmar que os controladores obtidos pelo método proposto responderam de forma eficiente ao distúrbio.

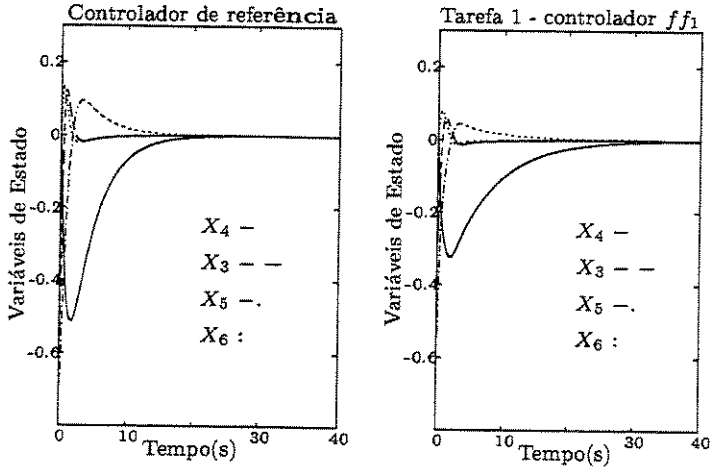

Figura 6.6: Respostas ao Impulso para o controlador de referência e para o controlador ff_1.

6.4.4 A Simulação Sequencial

A simulação sequencial consiste de uma busca que é realizada após o recebimento das populações permanentes de cada tarefa. A tarefa mestre seleciona os melhores indivíduos de cada população e monta uma nova população inicial que será submetida a um processamento sequencial. Para este caso, escolheu-se a estrutura de funções *fitness* ff_2. Este novo processamento tem dois objetivos, o primeiro é atingir alguma solução factível caso nenhuma das

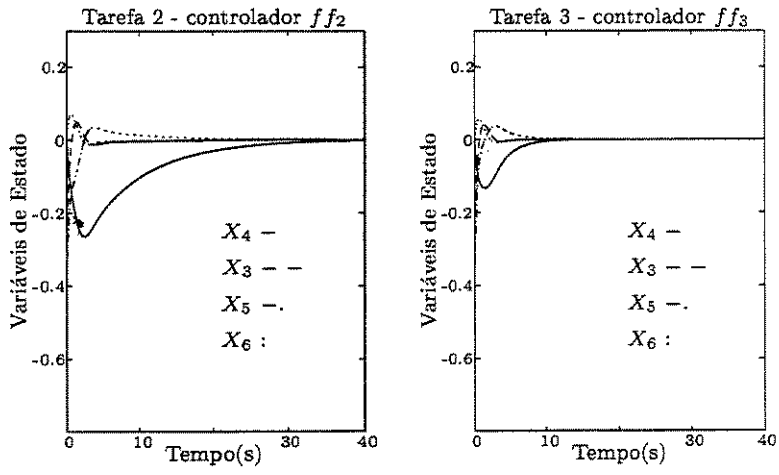

Figura 6.7: Respostas ao Impulso para os controladores ff_2 e ff_3.

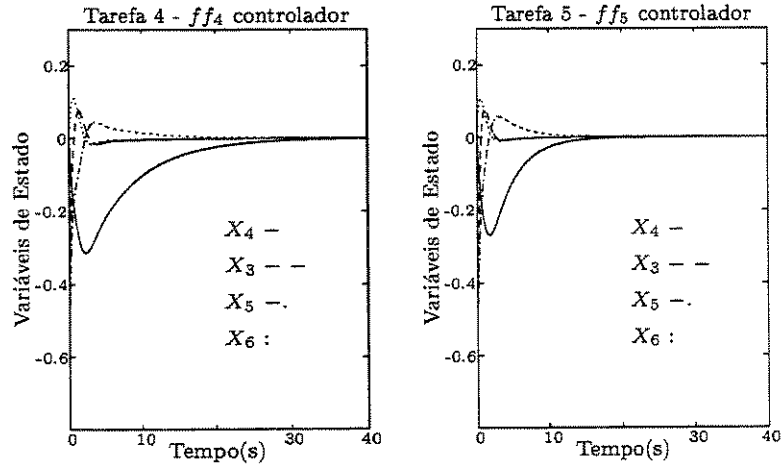

Figura 6.8: Respostas ao Impulso para os controladores ff_4 e ff_5.

tarefas tenha conseguido satisfazer todas as restrições de projeto e o segundo objetivo é promover melhoria do perfil da população permanente final. O comportamento do *ciclo de busca*, mostrando o perfil da população final, a pior sensibilidade dos autovalores e a função de custo do melhor indivíduo da população permanente, e o desempenho do sistema são apresentados no Apêndice (F).

6.4.5 Conclusões e Comentários

O desempenho do algoritmo *APAE-AGMO/RLQ* com a unidade de decisão (*UD*), que tem as suas estratégias fundamentadas no time de funções de *fitness*, no teorema do *schemata* e no paradigma *multiarmed bandit*, apresentou uma sensível melhoria em relação ao desempenho da implementação da subseção (6.3). O algoritmo teve a habilidade para determinar soluções de boa qualidade e o número de iterações gastas do *ciclo de busca* foi relativamente menor do que na implementação anterior, isto é, sem a atuação da *UD*.

O desempenho do otimizador-AG pode ser melhorado se a *UD* tem uma maior liberdade de interferir no comportamento da busca; esta afirmativa é verificada da seguinte maneira: para o algoritmo descobrir soluções factíveis o projetista deve fazer ajustes manuais na magnitude numérica dos indivíduos-QR e nos parâmetros multiplicativos, $q(t)_{idad}$ e $r(t)_{idad}$, da operação *crossover*, equações (3.9) e (3.10), que variam de acordo com a idade da população. Estas observações deram origem ao desenvolvimento de uma *UD* com certo grau de inteligência, na qual, a direção da busca na iteração $k+1$ é determinada de acordo com resultados obtidos na iteração k. As próximas seções apresentam os resultados que têm por base a inserção de estratégias que visam direcionar a busca; estas estratégias têm por base a indução de nichos, (Bottura e Fonseca Neto, 1999*f*), articulação de preferências, subseção (4.3.4), e o ajuste de parâmetros, subseção (4.3.5).

6.5 *UD* baseada em Regras

Apresenta-se a sistemática utilizada para determinar os ajustes de parâmetros do otimizador-AG e os resultados obtidos após a implementação de regras que definem ajustes de parâmetros da operação *crossover*.

A unidade de decisão (*UD*) foi utilizada para guiar a busca através de regras e estas são inseridas na *UD* após uma análise do comportamento da

busca pelo projetista.

O procedimento desenvolvido e verificado nesta subseção é o primeiro passo na direção do estabelecimento de uma metodologia de projeto inteligente para a alocação de autoestruturas.

6.5.1 A Unidade de Decisão Baseada em Regras

O conceito de unidade de decisão (*UD*) encontra-se bem definido em (Chankong e Haimes, 1983) e comentado no capítulo (4). Uma aplicação do conceito de unidade para tomada de decisão junto com técnicas de computação evolutiva foi desenvolvida por (Fonseca, 1995); sua *UD* baseia-se em *goals* e prioridades e as suas regras são formuladas em termos de operadores relacionais.

As estratégias aqui desenvolvidas, baseiam-se no teorema do *Schema* e no paradigma do *multiarmed bandit*, e são transformadas em regras *booleanas* e inseridas na unidade de decisão. Estas estratégias também são apresentadas em (Bottura e Fonseca Neto, 1999*c*).

As regras inseridas, chamadas de R_{idad} e Q_{idad}, atuam diretamente nos parâmetros da operação *crossover*, que são os fatores multiplicativos desta operação, conforme pode ser verificado no conjunto de equações (4.16) para os cromossomos do tipo-Q; para os do tipo-R as equações são similares. Estes parâmetros inicialmente foram projetados para variar de acordo com a idade da população e agora possuem uma segunda atribuição que é guiar a busca do otimizador-AG.

A unidade de decisão proposta, Figura (6.9), consiste de uma plataforma lógica baseada em regras, cuja principal função é formular decisões estratégicas que guiam a busca do otimizador-AG. As decisões estão fundamentadas em históricos que são indiretamente produzidos pelas operações genéticas e são formuladas através de cálculos realizados pelas estruturas do time de *fitness*, do teorema do *schemata*, do paradigma *multiarmed bandit* e da experiência do projetista. Nesta etapa, as decisões foram introduzidas interativamente para quebrar o conservantismo de regras estáticas e como consequência tem-se a definição de novas direções para a busca.

A estrutura básica da *UD* e o seu ponto de interação com o otimizador-AG são apresentadas na Figura (6.9). A regra R_{idade} é implementada na unidade de decisão e atua na formação de novos alelos-R através da operação *crossover*. As regras que têm por base o teorema do *schema* e o paradigma *multiarmed bandit* podem atuar nos alelos-Q e R.

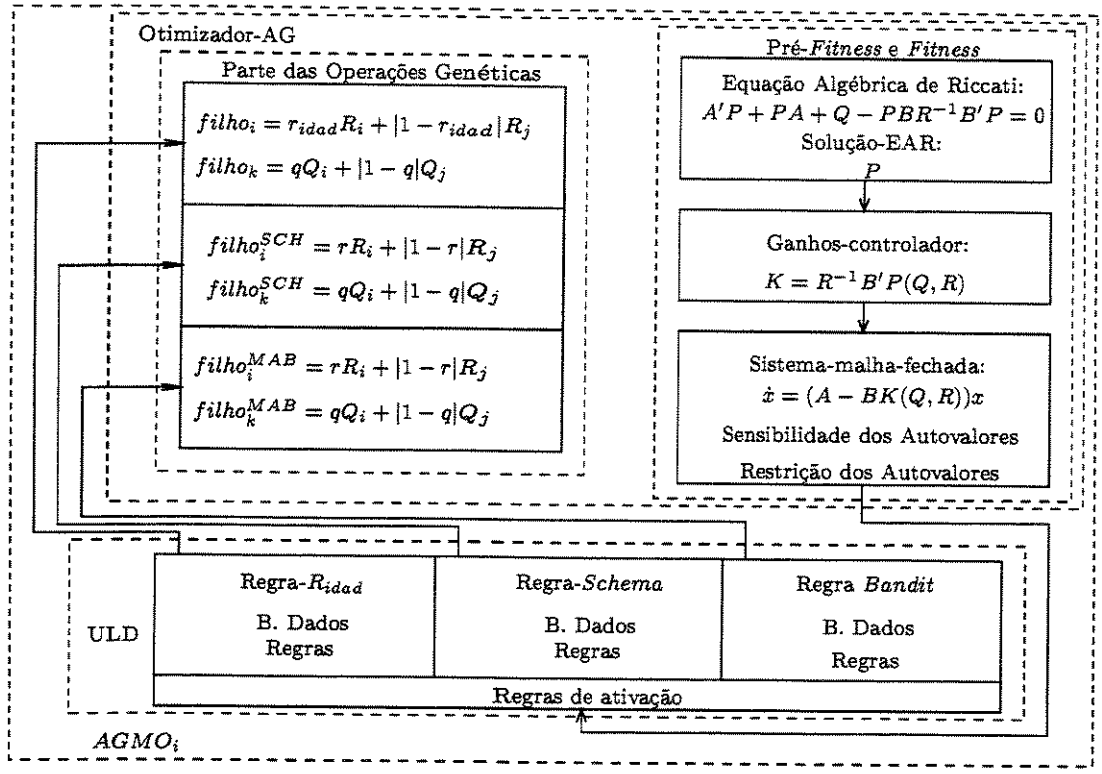

Figura 6.9: Interações entre *UTD* baseada em regras e o otimizador-AG em um $AGMO_i$.

Existem três condições para as regras serem disparadas, conforme estabelecido no capítulo (4), subseção (4.3.1), que são: pequena probabilidade de ocorrência, convergência prematura para soluções não factíveis da população-QR e deterministicamente entre vida e morte de gerações. No caso específico da regra R_{idad}, esta é disparada quando não existe convergência para uma determinada quantidade de iterações do *ciclo de busca* ou existe convergência prematura da população-QR.

6.5.2 Simulações

As simulações foram programadas para 8 tarefas paralelas e foram distribuídas em um ambiente computacional de alto desempenho. Uma das ta-

refas, coordenador ou *mestre*, criou aleatoriamente uma população inicial com dez indivíduos e estes foram enviados para as outras 7 tarefas *escravo*. Este ambiente baseia-se em máquinas *IBM-SP*, cujas características estão descritas no apêndice (B).

Para não perder a continuidade da apresentação do método proposto para o ajuste de parâmetros da operação *crossover* optou-se em apresentar este resultados preliminares e sua análise no Apêndice (G).

6.5.3 Conclusões e Comentários

A unidade de decisão, utilizando o teorema do *schemata*, o paradigma *multiarmed bandit* e o ajuste dos parâmetros da operação *crossover*, interagindo com o otimizador-AG, apresentou-se como um valioso dispositivo lógico para guiar o otimizador genético na procura de matrizes Q e R para o problema de alocação de autoestruturas via projetos RLQ.

O desempenho da unidade de decisão pode ser considerado bom, levando em consideração que um caso difícil de busca foi escolhido para verificar a eficiência do algoritmo *AGMOP-AAE/RLQ*. Uma possível melhoria para reduzir o tempo gasto na análise para tomar decisões e com tentativas para atuação das regras R_{idad} e Q_{idad}, objeto da exposição de resultados provenientes de pesquisa apresentados na seção (6.6), é o desenvolvimento de mecanismos para um ajuste adaptativo destes parâmetros.

6.6 Ajuste *on-line* de parâmetros

Esta seção apresenta os resultados computacionais do ajuste *on-line* de parâmetros da operação *X-over*. O desenvolvimento dos métodos para o ajuste encontra-se na subseção (4.3.5), capítulo (4), bem como em (Bottura e Fonseca Neto, 1999*d*). Os resultados expostos foram obtidos através de simulações realizadas em computadores de alto-desempenho; as características destas máquinas estão apresentadas no Apêndice (B), e o objetivo destas simulações é apresentar uma análise de desempenho do método proposto considerando a evolução das soluções e a resposta ao sinal impulso do sistema em malha fechada. Os resultados apresentados nestas subseções podem também ser encontrados em (Bottura e Fonseca Neto, 1999*e*).

6.6.1 Simulações - Ações do Ajuste

Neste item, apresenta-se uma análise comparativa da influência do ajuste dos parâmetros da operação cromossômica *crossover* no desempenho da busca realizada pelo *APAE-AGMO/RLQ*. Os resultados apresentados são provenientes de casos em que o algoritmo não teve a habilidade de fornecer soluções factíveis. Contudo, quando auxiliado pelas regras a alocação da autoestrutura tornou-se possível, determinando um controlador que satisfez as condições de projeto. A Tabela (4.2), subseção (4.3.5), apresenta as condições para variações destes parâmetros que estabelecem as diversas modalidades de regras.

Os perfis da população final da tarefa *mestre* para as regras quase-dinâmica, dinâmica determinística e adaptativa são apresentados na Figura (6.10). A regra quase-dinâmica não contribuiu para a busca de um controlador factível, Figura (6.10a). A regra adaptativa, Figura (6.10b) para $\Delta q(t) \neq 0$ e $\Delta r(t) \neq 0$ ajudou o otimizador-AG a determinar matrizes que conduziram a uma família de controladores factíveis, cujos valores das piores sensibilidades, Δ_j^{max}, $j = 1, \ldots, N$, estão entre 0.79 e 0.98. Contudo para duas variações destas regras, $\Delta q(t) = 0$ ou $\Delta r(t) = 0$, nenhum controlador factível foi determinado para um *ciclo de busca* de 100 iterações. A regra dinâmica determinística-direita, Figura (6.10c), melhorou a busca de indivíduos-QR, ($\Delta q(t) \neq 0$ e $\Delta r(t) \neq 0$) e ($\Delta q(t) \neq 0$ e $\Delta r(t) = 0$); seis controladores factíveis foram determinados para cada uma, cujos controladores Δ_j^{max} estão entre $0.90 - 0.97$ e $0.91 - 0.97$, respectivamente. Mas controladores factíveis não foram determinados para $\Delta q(t) = 0$ e $\Delta r(t) \neq 0$. A regra determinística-esquerda, Figura (6.10d), apresentou seis controladores para sua variação, $\Delta q(t) = 0$ e $\Delta r(t) \neq 0$, mas para as outras duas variações controladores factíveis não puderam ser determinados. Todos os tipos de regras e suas variações melhoraram o perfil de cada população final.

As Figuras (H.1-H.12), Apêndice (H), apresentam os resultados dos efeitos do ajuste de parâmetros, para um caso de díficil convergência, enfatizando variações dos parâmetros, maior sensibilidade do melhor indivíduo e sensibilidade média da população para um *ciclo de busca* de 150 gerações. Para cada regra, são realizadas 8 simulações paralelas. Cada figura representa os resultados da tarefa *mestre* e consiste de quatro curvas; curvas dos tipos a) e b) apresentam as variações dos parâmetros $q(t)_{idad}$ e $r(t)_{idad}$, respectivamente; as curvas dos tipos c) e d) apresentam a maior sensibilidade do melhor indivíduo para cada geração k, Δ_k^{max}, que é dada por $\min_j \max_i s_{ij}(Q, R)$, $i = 1, \ldots n$

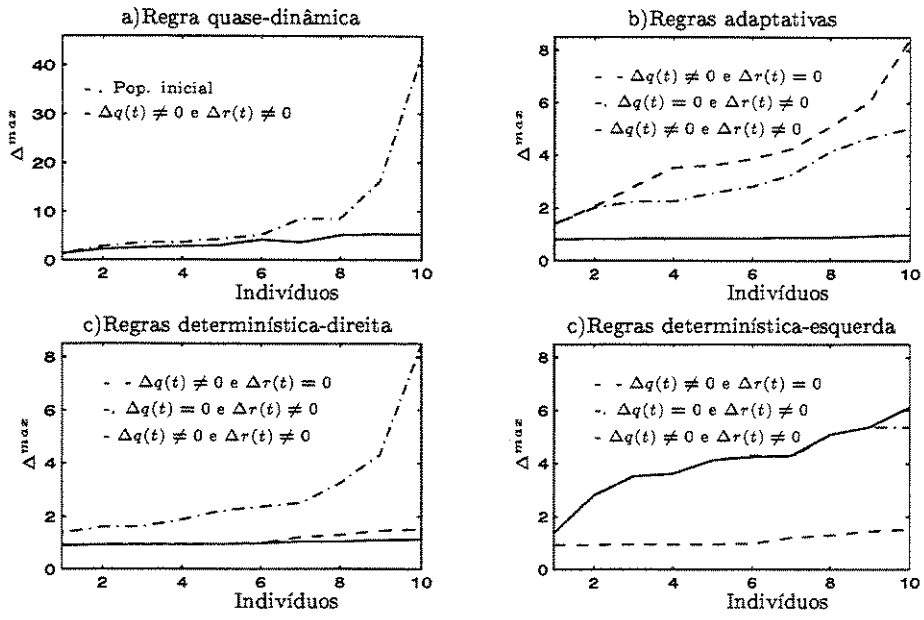

Figura 6.10: Ação do ajuste de parâmetros no perfil da população final da tarefa mestre.

e $j = 1, \ldots N$, onde n é a ordem do sistema dinâmico e N é a quantidade de indivíduos da população permanente, e a média da pior sensibilidade da população permanente, Δ_k^{med}, dada por $\sum_j^N \max_i s_{ij}(Q,R)/N$, respectivamente. Uma característica peculiar destes resultados: após o término da centésima quinta iteração, como pode ser observado nas curvas a) e b) das citadas figuras, as regras deixam de atuar, os controles dos ajustes dos parâmetros são retirados de ação e estes passam a sofrer variações aleatórias.

6.6.2 Simulações Exaustivas

As Tabelas (H.1-H.3), Apêndice (H), apresentam os resultados de simulações paralelas para 10 casos, considerando o mesmo sistema dinâmico e as mesmas condições operacionais. Para estas situações o *APAE-AGMO/RLQ* não apresentou soluções factíveis quando os parâmetros da operação *X-over* não sofrem qualquer ajuste dinâmico durante as iterações do *ciclo de busca*, mas soluções foram obtidas quando as regras dinâmicas foram ativadas para

$\Delta_q(t) \neq 0$ e $\Delta_r(t) \neq 0$. Oito processadores foram utilizados para cada caso, totalizando um total de 240 simulações no intuito de validar as melhorias advindas das regras propostas, que têm por objetivo direcionar as buscas das matrizes Q e R pelo otimizador-AG quando este é guiado pela unidade de decisão. É importante salientar que estas regras foram projetadas para guiar a busca do otimizador independentemente da escolha da população-QR inicial e sempre considerando uma pequena população permanente. Os tempos, típicos, gastos nos *ciclos de busca* para cada caso, considerando o processo *mestre* e as sete tarefas escravo, estão em torno de 26.90, 27.73, 27.79, 27.57, 33.60, 19.43, 26.44 e 26.30 segundos, respectivamemte.

6.6.3 Os Desempenhos dos Controladores

Os melhores controladores, para cada caso apresentado nas Tabelas (H.1-H.3), foram implementados no sistema dinâmico e aplicou-se um sinal impulso para verificar o desempenho de cada sistema de malha fechada. As respostas ao impulso foram comparadas com a do controlador de referência e observou-se que os controladores apresentaram desempenhos satisfatórios para este tipo de perturbação.

Apresenta-se os desempenhos de três controladores, obtidos com a ajuda da regra adaptativa e referentes ao caso 1, Tabela (H.1), comparados com o do controlador de referência, Figura (6.11). Observa-se que o controlador 5, Figura (6.11d) apresentou o melhor desempenho quando comparado com todos os outros controladores. O sistema dinâmico para os controladores mestre, Figura (6.11b), e 4, Figura (6.11c) apresentou um *overshoot* menor em relação ao controlador de referência, Figura (6.11a), mas apresentou um tempo de acomodação maior quando comparado com a resposta relativa à implementação do controlador de referência.

O Apêndice (H) mostra a resposta ao impulso do sistema dinâmico para uma implementação de quatro membros de um família de controladores.

6.6.4 Conclusões e Comentários

De uma maneira geral, conclui-se que as três regras apresentaram melhorias na busca do *APAE-AGMO/RLQ*, porque estas tiveram a habilidade de direcionar a busca para regiões factíveis do espaço de solução.

Verificou-se que a regra adaptativa apresentou uma solução com menos iterações, diferença não significativa dependendo da complexidade da função

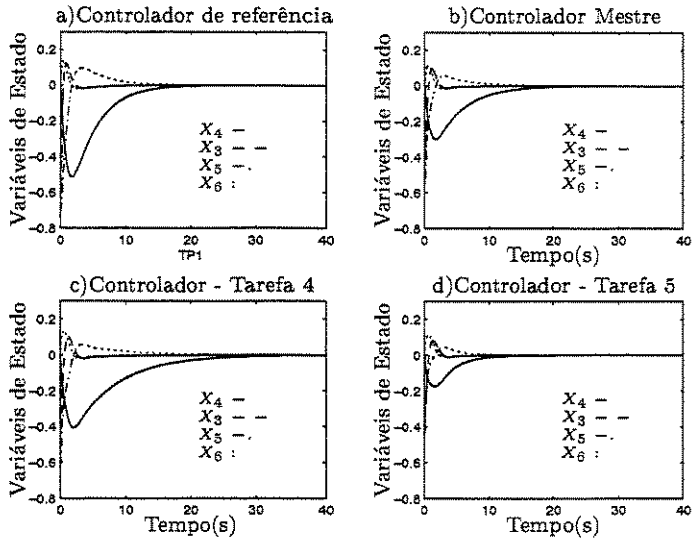

Figura 6.11: Caso I - Respostas ao Impulso - Controladores obtidos com a regra adaptativa

de *fitness*, e a regra determinística-direita apresentou mais soluções com um menor número de iterações e estas iterações foram em número menor que 100. A fim de retirar a máxima vantagem de cada regra, sugere-se para aumentar a eficiência do projeto: que as diferentes tarefas de um mesmo processo paralelo sejam guiadas no mínimo por uma das três regras, por exemplo: a Tarefa 1 é guiada pela regra adaptativa, a Tarefa 2 é guiada pela regra determinística-direita, a Tarefa 3 é guiada pela regra determinística-esquerda, e assim continua.

6.7 Análise da Autoestrutura

A resposta do sistema dinâmico ao sinal impulso, apresentada nas seções anteriores, é uma forma útil para analisar globalmente ao longo do tempo o comportamento do sistema dinâmico. Contudo, o comportamento dinâmico da resposta temporal do sistema pode ser melhor compreendido através da análise modal, que significa verificar os efeitos provocados pelas interações entre a autoestrutura, as condições iniciais e as matrizes de entrada e de

saída do sistema.

A determinação dos coeficientes de amortecimento, das frequências naturais amortecidas e das contribuições dos componentes dos vetores modais na entrada e na saída do sistema são os parâmetros utilizados para mostrar de forma direta a influência da autoestrutura na dinâmica do sistema.

Os autovalores reais e os seus autovetores associados são chamados de modos de primeira ordem e os autovalores complexos conjugados e seus autovetores associados são chamados de modos complexos ou de segunda ordem. A Tabela (6.5) apresenta as restrições de projeto que são: as faixas de autovalores e suas sensibilidades associadas. Estas restrições foram montadas a partir da autoestrutura fornecida pela implementação do controlador de referência no modelo linear do sistema dinâmico, equação (6.1).

No.	Autovalores	Sensibilidades
1	$-13.00 \leq Re \leq -23.00$	7.28
2	$-20.00 \leq Re \leq -30.00$	4.13
3	$-1.00 \leq Re \leq -3.00$ $3.000 \leq Imag \leq -3.00$	5.24
4	$-1.00 \leq Re \leq -3.00$ $3.00 \leq Imag \leq -3.00$	5.24
5	$-2.00 \leq Re \leq -3.00$	9.92
6	$-0.20 \leq Re \leq -3.00$	2.18

Tabela 6.5: Restrições de Projeto.

As Tabelas (6.6-6.7) apresentam a autoestrutura, a sensibilidade dos autovalores, os ganhos do controlador, os fatores de amortecimento e as frequências naturais de oscilação amortecidas para o sistema dinâmico. As informações destas tabelas são utilizadas para analisar a dinâmica do sistema do ponto de vista de autoestrutura.

A Tabela (6.6) mostra que os autovalores estão dentro da faixa de projeto e que as sensibilidades obtidas satisfazem as restrições especificadas.

Somente a análise dos autovalores não é suficiente para caracterizar o comportamento do sistema dinâmico. Sabe-se que cada componente do vetor de estado é formado pela contribuição de cada modo. Unindo os resultados apresentados na Tabela (6.6), autovalores, e na Tabela (6.7), autove-

No	Autoval.	Sens.	Amort.	Freq (rad/s)	Ganhos
1	-21.31	2.10			0.101 0.007 0.004 0.146 -0.017 -0.407
2	-26.94	3.55			-0.998 -0.886 -0.040 -1.577 1.536 3.084
3	-1.14 + j 1.27	3.63	0.67	1.71	
4	-1.14 + j -1.27	3.63	0.67	1.71	
5	-2.58	2.65			
6	-0.21	1.80			

Tabela 6.6: Autovalores, sensibilidades, coeficientes de amortecimento, frequências naturais de oscilação amortecidas e ganhos do controlador.

tores, verifica-se a influência de cada modo na resposta temporal do sistema dinâmico.

Cada autovalor da Tabela (6.6) está associado a um autovetor, Tabela (6.7), e cada componente de um autovetor é associado a somente um estado. Por exemplo, o modo de 1ª ordem, que corresponde ao autovalor 1 e ao autovetor 1 possui sua maior interação com o estado 1 porque seu maior elemento está na primeira posição, já para os estados 3 e 6 este modo não injeta contribuições. Observando a primeira linha da Tabela (6.7) verifica-se que os modos 2 e 6 não injetam contribuições no estado 1 e as contribuições dos modos de 2ª ordem, autovetores 3 e 4, são muito pequenas. Para os outros estados, a análise é realizada de forma similar.

Estado	Autovetores					
	1	2	3	4	5	6
1	-1.00	0.00	0.09 + j 0.26	0.09 - j 0.26	0.02	0.00
2	0.01	-1.00	0.73 - j 0.13	0.73 + j 0.13	0.78	0.00
3	0.00	0.00	-0.22 - j 0.03	-0.22 + j 0.03	-0.23	-0.98
4	-0.04	0.00	-0.25 + j 0.26	-0.25 - j 0.26	0.02	-0.04
5	0.02	-0.04	0.29 - j 0.24	0.29 + j 0.24	0.58	0.21
6	0.00	0.00	-0.21 - j 0.02	-0.21 + j 0.02	0.01	-0.03

Tabela 6.7: Autovetores associados aos autovalores da Tabela (6.6).

Como visto, na seção (2.2.1) os acoplamentos entre os modos e as saídas

e entre as entradas e os modos podem ser determinados pelos produtos CV e WB, respectivamente. Então, o produto dos autovetores pela matriz de saída C da equação (6.2) fornece o grau da distribuição dos efeitos dos modos sobre a saída, Tabela (6.8), e o produto dos autovetores recíprocos pela matriz B da equação (6.1) mostra o grau da distribuição dos efeitos do controle sobre os modos, Tabela (6.9).

As colunas da Tabela (6.8) representam vetores e os valores numéricos de seus componentes são interpretados como o grau de acoplamento entre os modos e a saída. Por exemplo, para o modo de 1ª ordem, o acoplamento representado pelo primeiro componente da coluna 6, é o que exerce maior influência na saída 1; os modos de 2ª ordem, colunas 3 e 4, exercem uma influência nesta saída similar à provocada pelo quinto modo de 1ª ordem e os primeiro e segundo modos são os modos que menos contribuem para a formação da saída. Uma análise similar pode ser realizada para as outras saídas. Verifica-se que a saída 4 é fortemente influenciada pelos modos de 2ª ordem.

Saída	Vetores					
	1	2	3	4	5	6
1	-0.001	0.002	-0.222 - j 0.033	-0.222 + j 0.033	-0.227	-0.976
2	-0.038	-0.001	-0.254 + j 0.255	-0.254 - j 0.255	0.022	-0.041
3	0.017	-0.043	0.295 - j 0.243	0.295 + j 0.243	0.585	0.210
4	-0.001	-0.000	-0.215 - j 0.021	-0.215 + j 0.021	0.012	-0.034

Tabela 6.8: O acoplamento modo-saída - CV.

A contribuição das entradas para os estados é observada através dos componentes dos vetores coluna da Tabela (6.9). Observa-se que a entrada 1 pode exercer uma grande influência no estado 1 porque o primeiro componente do vetor 1 possui um valor elevado quando comparado com os valores dos outros componentes. Contudo, a influência da entrada 2 neste estado pode ser muito pequena porque o componente do vetor 2 é bastante reduzido. A entrada 2 pode exercer uma grande influência no estado 2. Observando os componentes restantes do vetor de distribuição de entradas WB, verifica-se que as entradas exercem maior influência somente nos estados 1 e 2; para os outros estados a influência relativa à entrada é muito pequena.

Estado	Vetores	
	1	2
1	-20.705	0.034
2	-0.268	-26.768
3	0.133 + j 1.340	-0.092 + j 0.032
4	0.133 - j 1.340	-0.092 - j 0.032
5	-0.726	-2.080
6	0.216	0.483

Tabela 6.9: O acoplamento entrada-modo - WB.

6.7.1 Conclusões e Comentários

Esta análise mostrou numericamente o efeito da autoestrutura na resposta do sistema. Os autovalores determinam as taxas de decaimento ou de crescimento da resposta e os autovetores determinam a forma da resposta. A análise levantou diagnósticos sobre o comportamento dinâmico do sistema tendo por base o acoplamento modo-estado (autovetores), vetores de acoplamento modo-saída (produto CV) e vetores de acoplamento entrada-modo (produto WB). Obviamente, a tripla autovalores e autovetores à direita V e à esquerda W constitui os elementos básicos da análise.

A análise da autoestrutura apresentada nesta seção é uma ferramenta que pode ser utilizada tanto a *priori*, no sentido de levantar um diagnóstico de como a autoestrutura atual está influenciando a dinâmica do sistema e a partir deste ponto definir os requisitos de projeto, como a *posteriori* para verificar a satisfabilidade dos requisitos de projeto.

Uma análise do produto dos autovetores recíprocos pelas condições iniciais é útil para verificar a influência dos modos na resposta livre do sistema. O grau de acoplamento entrada-saída também pode ser diagnosticado através do produto dos vetores de acoplamento CV e WB, (Littleboy e Nichols, 1998).

Capítulo 7

Conclusões e Trabalhos Futuros

Esta tese apresentou um método para alocação computacional inteligente de autoestruturas para controle multivariável. O problema da alocação de autoestruturas para sistemas lineares dinâmicos foi abordado tanto no nível de uma nova formulação como no nível de uma solução que se adequa à formulação proposta.

A formulação proposta para o problema de alocação de autoestruturas teve como base o controle ótimo, especificamente o projeto do regulador linear quadrático (RLQ), e a otimização multiobjetivo. O problema de otimização multiobjetivo é montado a partir de índices de desempenho e de restrições que representam as autoestruturas calculadas e que são funções de controladores por realimentação de estado fornecidos via projeto do regulador linear quadrático.

O método de solução proposto tem por finalidade fornecer as matrizes de ponderação do projeto RLQ para que este determine o ganho do controlador de malha fechada que impõe a autoestrutura especificada. Este método teve por base uma técnica de inteligência computacional, especificamente o algoritmo genético, que realiza uma busca aleatória, polarizada e guiada, no espaço de solução para estas matrizes.

Objetivando reduzir o tempo de busca, desenvolveu-se um algoritmo genético com pequenas populações e um operador que tem como objetivo aumentar a diversidade genética. Para retirar um maior proveito de cada iteração do *ciclo de busca*, projetou-se uma unidade de decisão que implementa uma série de regras alimentadas pelas medidas de *fitness* e estas regras têm por finalidade guiar a busca. Um time de funções de *fitness* foi construído com o intuito de criar diferentes representações do mesmo objetivo; isto é

uma forma de atrair soluções que não conseguem otimizar uma dada representação para uma dado objetivo. Explorou-se também técnicas de indução de nichos, articulação de preferências, critério de otimalidade de *Pareto*, teorema do *schemata*, paradigma do *multiarmed bandit* e ajustes de parâmetros da operação *crossover* dentro de um contexto para reduzir o tempo de obtenção de uma solução ou para garantir uma solução factível. O critério da otimalidade de *Pareto* foi utilizado para não perder as melhores reduções das funções de sensibilidade dos autovalores ao longo da busca e com isto garantir um melhor condicionamento à autoestrutura alocada.

O algoritmo genético, que realiza a busca, junto com a unidade de decisão, que guia a busca, é chamado de algoritmo genético multiobjetivo ($AGMO$) para alocação de autoestruturas via projeto RLQ. Nesta fase, o algoritmo genético (AG) é chamado de otimizador genético porque suas características intrínsecas de inteligência computacional foram superadas pela inteligência computacional da unidade de decisão, que é a inteligência do algoritmo, e pela formulação de regras através de operadores relacionais e de funções *booleanas*, que são alimentadas pelas medidas de *fitness* fornecidas pelo AG.

O algoritmo multiobjetivo foi implementado utilizando o paradigma *SPMD* de paralelização e as bibliotecas para programação **PVM** e **MPI** que têm por base o paradigma *message passing*. O algoritmo foi paralelizado tendo dois objetivos: o primeiro objetivo: a paralelização do algoritmo genético, onde cada tarefa paralela parte do mesmo ponto inicial, realiza buscas independentes e no final seleciona as melhores soluções de cada processamento; o segundo objetivo: o processamento simultâneo de vários projetos RLQ.

Como metodologia de projeto, o algoritmo proposto é muito versátil pelas seguintes razões: a formulação permite a inclusão de vários objetivos ou restrições sem sofrer grandes modificações na sua estrutura básica; no paradigma *SPMD* é suficiente modificar um único programa-fonte, na implementação do paradigma de programação paralela *message passing* é suficiente a inclusão de funções *send-receive* para o caso da biblioteca **MPI**. Em relação ao ambiente computacional, o processamento pode ser realizado em computadores de alto-desempenho, em redes de estações de trabalho ou em computadores pessoais.

Devido à natureza computacional da proposta, exaustivas simulações foram realizadas para verificar o desempenho do método para alocação computacional inteligente de autoestrutura para controle multivariável. Estas simulações mostram que a formulação do problema, o método de solução e a sua implementação são de boa qualidade para o cumprimento dos seus

objetivos.

7.1 Trabalhos Futuros

Os tópicos listados a seguir são sugeridos como futuros objetos de pesquisa, a fim de promover a continuidade da pesquisa inicialmente desenvolvida *ad hoc* e apresentada nesta tese:

- Inclusão de outras funções objetivo no *time*, tais como funções que considerem a alocação dos autovetores recíprocos ou que considerem aspectos de robustez.
- Explorar o problema da alocação de zeros.
- Explorar variações no paradigma do algoritmo *APAE-AGMO/RLQ* de forma que as buscas das tarefas não sejam totalmente independentes, ocorrendo uma migração entre indivíduos de processamentos diferentes durante as iterações do *ciclo de busca.*
- Investigar o desempenho de controladores genéticos que reagem de forma dinâmica a perturbações na planta.
- Analisar a estabilidade dos controladores genéticos durante as primeiras iterações do *ciclo de busca*, tendo em vista aplicações *on-line* do método proposto.
- Investigar a solução do problema *RLQ* utilizando outras técnicas de computação evolutiva.
- Desenvolver um algoritmo que explore uma paralelização da equação algébrica de *Ricatti* (*EAR*), isto é, uma paralelização no nível da estrutura algébrica desta equação. A Figura (7.1) esboça uma possível implementação deste algoritmo. Neste caso, tem-se dois tipos bem definidos de máquinas, que são: as máquinas *RLQ*, responsáveis pelas tarefas de geração das matrizes de peso, e as máquinas *DEEAR*, responsáveis pelas tarefas para solucionar a decomposição da estrutura algébrica de cada *EAR*. Com isto espera-se reduzir sensivelmente a carga computacional das máquinas *RLQ* para projeto de controladores e simulação de sistemas de grande porte. Esta implementação

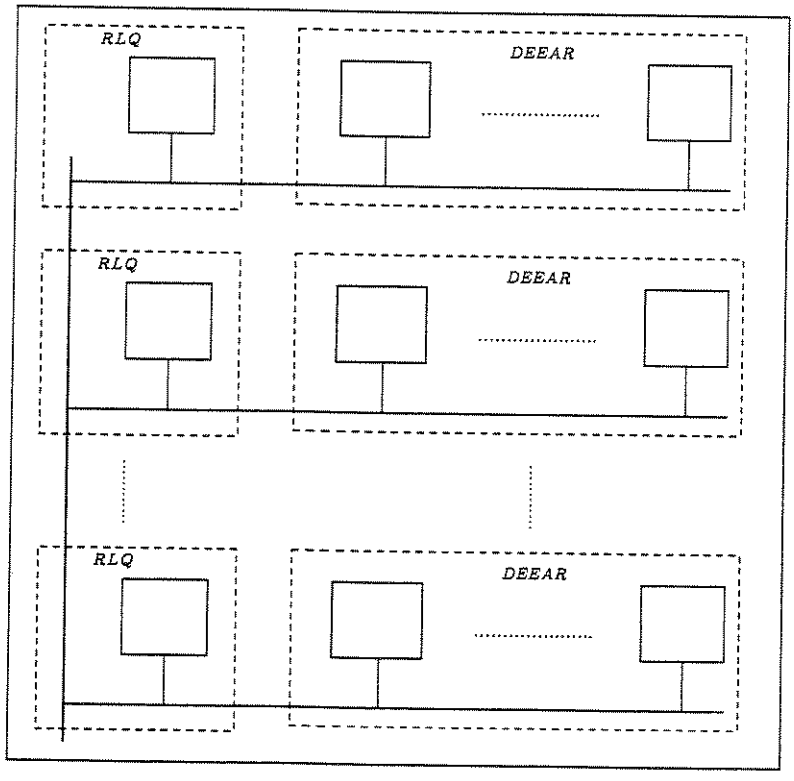

Figura 7.1: Esquema para Paralelização do Algoritmo *PMOGA-RLQ* em dois níveis.

pode ser feita utilizando o método desenvolvido por (Tamariz, 1999), e sintetizado em (Bottura *et al.*, 1999*b*) e (Bottura *et al.*, 1999*a*).

- Aplicar os resultados desta pesquisa no projeto de controladores multivariáveis nas mais variadas áreas.

Apêndice A

Solução do Problema RLQ

Utilizando o princípio de otimalidade obtem-se a equação de otimização de *Hamilton-Jacobi* para o problema do regulador linear quadrático definido pelas equações (2.86) e (2.87).

Dado um sistema dinâmico não linear, equação (A.1), o principal objetivo do problema de controle ótimo consiste na determinação de uma lei de controle por realimentação de estado $u = \phi(x,t)$ que minimize o índice de desempenho dado pela equação (A.2) e tendo como restrição o próprio sistema:

$$\dot{x} = f(x,u,t) \tag{A.1}$$

onde $f(x,y,u,t)$ caracteriza o sistema dinâmico contínuo no tempo t como uma função do estado x e da lei de controle u.

$$J_1 = m\left[x(T)\right] + \int_t^T l(x,u,\tau)d\tau \tag{A.2}$$

onde $l(x,u,\tau)$ é a função de perdas, $m(x)$ é o custo terminal e T é o horizonte de otimização.

Define-se o valor mínimo do índice de desempenho (A.2) para a lei de controle $u[t,T]$ no intervalo $[t,T]$, tendo como instante inicial t e o estado inicial $x(t)$, por:

$$J_1^*(x,t) = \min_{u[t,T]} \left\{ m[x(T)] + \int_t^T l(x,u,\tau)d\tau \right\} \tag{A.3}$$

Aplicando o princípio da otimalidade na equação (A.3), ie, o valor mínimo

de J_1 a partir de $x(t+\Delta t)$, no instante $(t+\Delta t)$, é dado por $J_1^*[x(t+\Delta t), t+\Delta t]$, obtem-se a seguinte relação para $J_1^*(x,t)$:

$$J_1^*(x,t) = \min_{u[t,t+\Delta t]} \left\{ J_1^*[x(t+\Delta t), t+\Delta t] + \int_t^T l(x,u,\tau)d\tau \right\} \tag{A.4}$$

Uma observação importante, (Dorato *et al.*, 1995), é que a determinação de uma lei de controle no intervalo $[t,T]$ foi reduzida à determinação de uma lei de controle no intervalo reduzido $[t,t+\Delta t]$. Manipulando a equação (A.4), aproximando o termo integral por $l(x,u,t)\Delta t$ e expandindo $J_1^*[x(t+\Delta t), t+\Delta t]$ em série de *Taylor* multivariável no ponto $(x(t),t)$, onde $x(t+\Delta t)-x(t)$ é aproximado por $f(x,u,t)\Delta t$, tem-se:

$$J_1^*(x,t) = \min_{u(t)} \left\{ l(x,u,t)\Delta t + J_1^*(x,t) + \frac{\partial J_1^*}{\partial t}\Delta t + \left[\frac{\partial J_1^*}{\partial x}\right]' f(x,u,t)\Delta t + o(\Delta t) \right\} \tag{A.5}$$

onde $\frac{\partial J_1^*}{\partial x}$ representa o gradiente de J_1^* com respeito ao vetor de estado x e $o(\Delta t)$ representa os termos de alta ordem em Δt.

A equação de otimização de *Hamilton-Jacobi* é obtida quando $\Delta t \to 0$ na equação (A.5). Então,

$$-\frac{\partial J_1^*}{\partial t} = \min_{u(t)} \left\{ l(x,u,t) + \left[\frac{\partial J_1^*}{\partial x}\right]' f(x,u,t) \right\} \tag{A.6}$$
$$J_1^*(x,T) = m(x), \qquad \forall x$$

A solução da equação de otimização (A.6) é obtida quando efetua-se a sua minimização, equação (A.7), substitui-se a lei de controle na equação de otimização e resolve-se a equação diferencial parcial (A.8) para $J_1^*(x,t)$, sujeita a condição de contorno $J_1^*(x,T) = m(x)$:

$$u^* = \psi\left(\frac{\partial J_1^*}{\partial x}, x, t\right) \tag{A.7}$$

$$-\frac{\partial J_1^*}{\partial t} = l(x,u,t) + \left[\frac{\partial V^*}{\partial x}\right]' f(x,\psi,t) \tag{A.8}$$

Uma vez de posse da solução $J_1^*(x,t)$ calcula-se o seu gradiente e obtém-se a lei de controle ótimo por realimentação de estado:

$$u^* = \psi\left(\frac{\partial J_1^*}{\partial x}, x, t\right) = \phi(x,t) \tag{A.9}$$

Substituindo $l(x,u,t) = x'Qx + u'Ru$, $f(x,u,t) = Ax + Bu$ e $m(x) = x'Mx$ na equação (A.6) retorna-se ao problema RLQ. Então,

$$-\frac{\partial J^*}{\partial t} = \min_{u(t)}\left\{x'Qx + u'Ru + \left[\frac{\partial J^*}{\partial x}\right]' Ax + Bu\right\} \tag{A.10}$$
$$J^*(x,T) = x'Mx, \qquad \forall x$$

A minimização da equação (A.10) com relação a u conduz a minimização do problema RLQ. A minimização é efetivada quando o gradiente torna-se um vetor nulo e a referida equação é resolvida para u. Então,

$$u^* = -\frac{1}{2}R^{-1}B'\frac{\partial J^*}{\partial x} \tag{A.11}$$

Com relação a forma de J^*, supõe-se que esta também seja quadrática, equação (A.12); esta consideração é razoável já que uma forma integral quadrática é avaliada no estado inicial do sistema:

$$J^* = x'P(t)x \tag{A.12}$$

onde $P(t)$ é simétrica. Substituindo a equação (A.12) e o seu gradiente na equação diferencial parcial (A.7) tem-se a seguinte equação:

$$-x'\dot{P}x = x'\left[A'P + PA + Q - PBR^{-1}B'P\right]x \tag{A.13}$$

e sua condição de contorno é dada por:

$$x'P(T)x = x'Mx \tag{A.14}$$

A equação diferencial matricial de *Riccati*, equação (A.15), e seu valor de contorno, equação (A.16) são obtidos das equações (A.13) e (A.14) sabendo-se que estas relações são válidas para todo x:

$$-\dot{P} = A'P + PA + Q - PBR^{-1}B'P \tag{A.15}$$

$$P(T) = M \tag{A.16}$$

Agora, a solução da equação de *Riccati* $P(t)$ fornece a lei do controle ótimo por realimentação de estado. Então, substituindo o gradiente da equação (A.11) na lei do controle, equação (A.12), obtém-se a forma analítica para u^*:

$$u^* = -R^{-1}B'P(t)x \tag{A.17}$$

Um caso particular do problema *RLQ* ocorre quando as matrizes do sistema dinâmico (A, B e C) e as matrizes de ponderação (Q, R e M) são invariantes no tempo e o intervalo de otimização é infinito. Pode-se mostrar que a equação diferencial matricial de *Riccati*, equação (A.15), transforma-se na equação algébrica matricial de *Riccati* (*EAMR*):

$$0 = A'P + PA + Q - PBR^{-1}B'P \tag{A.18}$$

A solução desta equação quando existir é invariante no tempo e pode ser obtida por diversos métodos, tais como: *Newton*, função *sign* matricial, autosistemas e métodos simpléticos, (Petkov *et al.*, 1991). Nesta tese utilizou-se o método de *Schur*, que poder ser classificado como um método de autosistemas, desenvolvido por (Laub, 1979). O problema *RLQ* para estas condições é chamado de problema do regulador em regime permanente e a lei do controle ótimo é dada por:

$$u^* = -R^{-1}B'\bar{P}x \tag{A.19}$$

onde $\bar{P}$ é a solução da equação (A.18).

As condições de existência e estabilidade para a solução do problema do regulador em regime permanente são garantidas para $M = 0$, $R > 0$ e $Q = D'D$, tal que o par (A, B) seja controlável e o par (A, D) seja observável, (Laub, 1979). Estas condições garantem uma solução definida positiva única para a equação (A.18), $\bar{P}$, e que o sistema de malha fechada com realimentação ótima através da lei de controle dada pela equação (A.19) seja assintoticamente estável.

A vantagem de utilizar o problema *RLQ* no projeto de controladores para alocação de autoestruturas, em relação a outros métodos de projeto, é que o método *RLQ* garante intrinsicamente certas propriedades de robustez, tais

como: margem de ganho infinitamente crescente e margens de fase entre $\pm 60^0$.

Apêndice B

Ambientes Computacionais

As topologias e as características dos ambientes paralelos utilizados para verificar o desempenho do algoritmo paralelo *AGMOP-AE/RLQ* nos ambientes distribuído paralelo e exclusivamente paralelo, conforme definido no capítulo (5), subseção (5.4.2), são apresentadas neste apêndice.

B.1 O Ambiente Paralelo Distribuído

O ambiente paralelo distribuído é formado por 10 estações de trabalho *SUN* que são vistas como uma máquina paralela virtual. Os seguintes modelos de estações fazem parte da rede: *SPARC-20, SPARC-5 ULTRA, SPARC-4, SPARC-Classic* e *SPARC-2.* Esta rede pertence ao *DMCSI* e sua configuração está apresentada na Figura (B.1). Este rede é constituída de duas subredes que possuem como ponto de ligação (*Gateway*) entre elas a máquina *Papagaio.* As simulações foram realizadas na subrede *LCI*, utilizando as máquinas *Águia*, que é a servidora desta rede e é o ponto de ligação com a rede da *FEEC, Saturno, Urano* e das outras máquinas.

B.2 O Ambiente Paralelo de Alto Desempenho

O computador paralelo do ambiente de alto desempenho é um *IBM 9076 SP2* e encontra-se instalado no *Centro Nacional de Processamento de Alto Desempenho em São Paulo* (*CENAPAD-SP*). Segundo informações obtidas em (Steen, 1997), este computador possui as seguintes características:

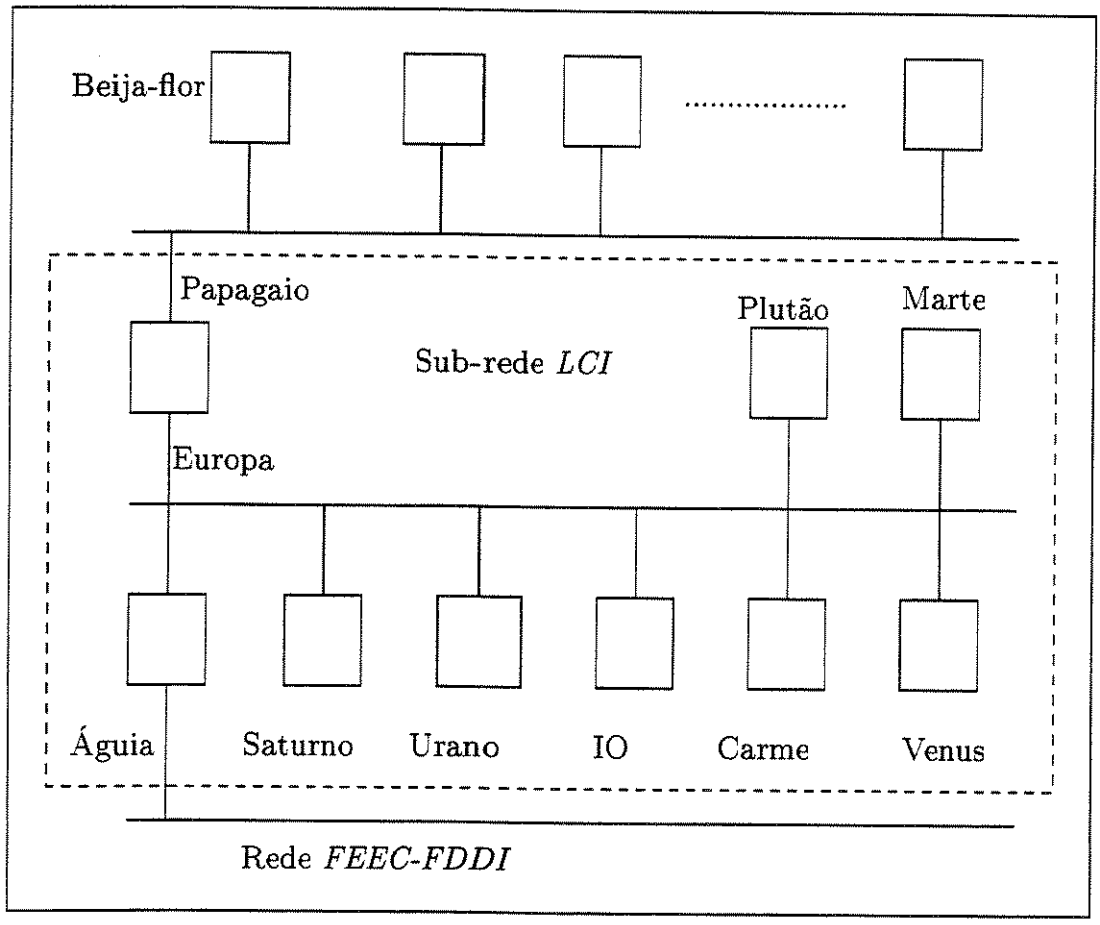

Figura B.1: Arquitetura da rede *DMCSI*.

1. Tipo de Máquina: *RISC, cluster* de memória distribuída e multi-processador;
2. Modelo: *IBM 9076 SP2*;
3. Sistema operacional: *AIX*, variação do *UNIX*;
4. Estrutura de Conexão: Dependente do tipo de Conexão;
5. Compiladores: *XL Fortran, XL C, XL C++*.

A seguir apresenta-se breve descrição da solução *IBM RS/6000* envolvendo a arquitetura *MPP* e *SMP*; esta descrição é uma reprodução da descrição que encontra-se na *homepage CENAPAD-SP*.

O sistema *SP* permite o uso simultâneo de dúzias a centenas de processadores *RISC* para um problema computacional. A base do *SP* é o nó processador. Ele consiste de um microprocessador *P2SC* ou de um multimicroprocessador simétrico *PowerPC* (*SMP*), memória, slots de expansão *PCI* ou Micro *Channel* para *I/O* e conectividade, e dispositivos de disco. Os três tamanhos de nós (*thin, wide* e *high*) podem ser combinados em um sistema, e são montados em *frames short* (baixos) ou *tall* (altos). Dependendo dos tamanhos dos nós usados, um *frame SP tall* pode conter até dezesseis nós. Estes *frames* podem ser interconectados para formar um sistema com até 128 nós (512 através de requisição especial) onde um máximo de 64 *high nodes SMP* podem ser instalados por sistema.

Computação paralela efetiva requer comunicação entre nós de alta velocidade e baixa latência. O *SP Switch* provê uma taxa máxima de transferência de dados bi-direcional de mais de 120 MB/segundo entre cada par de nós. O *SP Switch Router* é um *gateway* de alto desempenho que provê o mais rápido nível de comunicação entre o sistema *SP* e o mundo externo, ou entre múltiplos sistemas *SP*. Este *gateway SP* combina o *Ascend GRF* com o *IBM SP Switch Router Adapter* para permitir conexão direta de rede com o *SP Switch*. Outras placas conectam-se a uma variedade de redes externas padrões. Cada placa tem seu próprio *hardware*, permitindo o crescimento de *I/O* do *SP* para até um-a-um com o número de placas.

O sistema *SP* pode também crescer em *I/O* de disco para até um-a-um com processadores e memória, possibilitando acesso a terabytes de dados e facilitando o gerenciamento de expansões e *upgrades*.

O *IBM 9076 SP* instalado no *CENAPAD-SP* possui três frames, que representam cerca de 22 GFLOPs de capacidade de processamento total. Contudo só um dos *frames* foi utilizado.

Agora, descreve-se a arquitetura do *frame* utilizado. A arquitetura *MMP* ou de processamento massivamente paralelo é *shared nothing*. Os processadores são fracamente acoplados e conectados através de um *link* de alta velocidade. Cada processador ou nó possui sua própria memória, sistema operacional ou disco. Como produto *IBM* a máquina paralela é chamada de *SP* ou *scalable power parallel*.

O frame utilizado possui a seguinte configuração:

1. 16 nós com processadores *IBM/RS6000 Power2 Super Chip*, de 120 MHz,

2. Desempenho teórico de 480 MFLOPs por nó, totalizando 7,6 GFLOPs,

3. 512 MB de memória *RAM* por nó, totalizando 8 GB,
4. 4,5 GB de espaço em disco por nó, totalizando 72 GB (/work com 2 GB),
5. 50 GB em discos externos (*SSA*),
6. Comunicação entre nós e com os servidores de arquivos:
   - Interface *FDDI* - 12.5 Mb/s (100 Mbps),
   - Interface *SPS* - 150Mb/s bi-direcional,
   - Hostnames: sp1s01.cna.unicamp.br a sp1s16.cna.unicamp.br.

Apêndice C

Primitivas das Bibliotecas PVM e MPI

Neste apêndice descreve-se as características das primitivas das bibliotecas **PVM** e **MPI** utilizadas na paralelização do algoritmo *AGMOP-AE/RLQ*.

C.1 PVM

Bob Manchek, no *PVM-FAQ*, define o **PVM** (*Parallel Virtual Machine*) como um sistema de programação baseado em *message-passing*, projetado para unir máquinas *hosts* formando uma máquina virtual, a qual constitui uma única fonte computacional gerenciável. A máquina virtual pode ser formada de vários tipos de *hosts*, em localizações fisicamente remotas. As aplicações **PVM** podem ser constituídas de processos separados ou componentes escritas em *C*, C^{++} e *FORTRAN*. O sistema é portável para uma grande variedade de arquiteturas, incluindo *workstations*, multiprocesadores, supercomputadores e *PC's*.

C.1.1 Primitivas do PVM

Aqui descreve-se as primitivas para controle do processo, informativas, manipulação com *buffers*, envio e recebimento de mensagens:

1. Controle do Processo

   call pvmfmytid(mtid)

Objetivo - retorna o *tid* do processo.

Argumento:

- *tid* - variável inteira que retorna o identificador da tarefa que invoca o processo **PVM**.

Descrição - Esta rotina adiciona o processo no **PVM** na sua primeira chamada e gera um *tid* único se o processo não foi criado por uma chamada *pvm − spawn*. Se o **PVM** não foi inicializado antes de uma aplicação chamada desta rotina retornará $tid < 0$.

call $pvmfspawn(task, flag, where, ntask, tids, numt)$

objetivo - inicializa um novo processo **PVM**.

Argumentos:

- *task*− string de caracteres contendo o nome do arquivo executável do processo **PVM** para ser inicializado,
- $flag$− inteiro especificando as opções de *spawn*,
- $where$− string de caracteres especificando onde começar o processo **PVM**. Dependendo do valor de $flag$, $where$ pode ser o nome de um *host* ou uma classe de arquitetura **PVM**,
- $ntask$− inteiro especificando o número de cópias de código executável para ser inicializado,
- *tids*− array inteiro de comprimento no mínimo $ntask$. No retorno o array contém os *tids* do processo **PVM** inicializado.
- *numt*− inteiro que retorna a quantidade de tarefas inicializadas. Valor menor que zero indica um erro do sistema.

call $pvmfexit(info)$

objetivo - informa ao *pvmd* que este processo está deixando o **PVM**.

Argumento:

- $info$− código de *status* inteiro de retorno da rotina. Valor menor do que zero indica erro.

2. Informação:

   call pvmfparent(tid)

   Objetivo - retorna o *tid* do processo que *spawned* o processo.

   Argumento:

   - *tid*– inteiro que retorna o indentificador da tarefa-pai do processo que invoca esta rotina. Se o processo não foi criado com a *pvmfspawn*, o $tid = PvmNoParent$.

3. Mensagens em buffers:

   call pvmfinitsend(encoding, bufid)

   Objetivo - Limpa o buffer default de envio e especifica a codificação da mensagem.

   Argumentos:

   - *encoding*– inteiro especificando o próximo esquema de codificação de mensagem
   - *bufid*– inteiro retornando a mensagem do identificador do *buffer*. Valor menor que zero indica um erro.

4. Envio:

   call pvmfpack(what, xp, nitem, stride, info).

   Objetivo - Empacota o *buffer* ativo de mensagens com *arrays* de um tipo de dado.

   Argumentos:

   - *what*– inteiro especificando o tipo de dado que está sendo empacotado.
     (a) STRING REAL4 REAL8 BYTE1
     (b) INTEGER2 INTEGER4 COMPELEX8 COMPELEX16
   - *xp*– apontador para o começo de um bloco de *bytes*. Pode ser qualquer tipo de dados, mas deve casar com o tipo correspondente de desempacotamento,

- *nitem*–quantidade total de itens a serem empacotados, não o número de *bytes*,
- *stride*– o *stride* que dever usado quando empacota-se os ítens. Para números complexos especificar 2, caso contrário especificar 1,
- $info$– código de *status* inteiro. Valor menor que zero indica erro.

call $pvmfsend(tid, msgtag, rcode)$.

Objetivo - Envia os dados no *buffer* de mensagem ativo.

Argumentos:

- *tid*– identificador da *task* inteiro do processo destino,
- $msgtag$–tag da mensagem inteiro fornecido pelo usuário, o $msgtag \geq 0$,
- $info$–código de *status* inteiro retornado pela rotina. Valor menor do que zero indica um erro.

5. Recebimento:

   call $pvmfrecv(mtid, mtype, rcode)$.

   Objetivo - Receber uma mensagem.

   Argumentos:

   - *tid*– identificador de tarefa inteiro do processo de envio fornecido pelo usuário. Um valor -1 neste argumento casa com qualquer *tid*,
   - $msgtag$– tag da mensagem inteiro fornecido pelo usuário,$msgtag$ deve ser maior ou igual a zero. Um valor -1 neste argumento casa com qualquer *tag* de mensagem,
   - $bufid$–inteiro que retorna o identificador do novo *buffer* ativo de recebimento; valor menor do que zero indica um erro.

   call $pvmfunpack(what, xp, nitem, stride, info)$.

   Objetivo - Desempacota o *buffer* ativo de mensagem em *arrays* do tipo de dado especificado.

   Argumentos:

- *what*– inteiro especificando o tipo de dado que está sendo desempacotado. Opções:
  (a) STRING REAL4 REAL8 BYTE1
  (b) INTEGER2 INTEGER4 COMPELEX8 COMPELEX16
- *xp*–apontador para o começo de um bloco de *bytes*. Pode ser qualquer tipo de dados, mas deve casar com o tipo correpondente de empacotamento,
- *nitem*–quantidade total de itens a serem desempacotados, não o número de bytes,
- *stride*– usado quando empacotou-se os ítens. Para números complexos especificar 2, caso contrário especificar 1,
- *info*–código de *status* inteiro. Valor menor que zero indica erro.

C.2 MPI

A linguagem de programação **MPI** é o resultado do esforço conjunto dos fabricantes de computadores, laboratórios do governo dos Estados Unidos e pesquisadores das universidades para desenvolver uma interface padrão para o paradigma de *message passing* de programação paralela. Em 1994 foram apresentadas as especificações do padrão **MPI**; este padrão define o núcleo, a sintaxe e a semântica das rotinas da biblioteca *message-passing* que podem ser implementadas em arquiteturas paralelas MMP.

Portabilidade, eficência e robustez são as principais características do **MPI**. A Portablidade significa que uma aplicação **MPI** só necessita recompilação para utilização de uma implementação diferente. Eficiência significa que as implementações do **MPI** devem interagir eficientemente com bibliotecas nativas de *message passing*. Robustez significa que o **MPI** fornece recursos significantes para o desenvolvimento de bibliotecas paralelas.

Um conjunto de processos e um meio de comunicação lógica conectando estes processos forma um programa **MPI**. O modelo de memória do **MPI** é logicamente distribuído e a comunicação entre processos necessita a chamada de rotinas em ambos processos.

As rotinas de comunicação *ponto-a-ponto* são as rotinas de comunicação mais importantes do **MPI** porque permitem que os processos troquem informações cooperativamente. O *message passing* consiste desta forma cooperativa de comunicação entre processos.

Diferente do **PVM** um conjunto de apenas 6 subrotinas é suficiente para paralelizar um algoritmo no **MPI**. A próxima subseção descreve estas 6 subrotinas que são utilizadas para inicializar o processo, identificar processos e **MPI**, contar processos, enviar mensagens, receber mensagens e finalizar processos no **MPI**. O texto apresentado nesta seção e as primitivas apresentadas na subseção (C.2.1) tem como referências (CENAPAD, 1996), (Gropp *et al.*, 1994) e (Geist *et al.*, 1994).

C.2.1 Primitivas do MPI

Esta seção descreve as 6 primitivas no **MPI** que são suficientes para implementar programas paralelos. Então:

1. Inicializar o processo -

   `MPI_INIT`

   - Características:
     - Primeira rotina de cada processo,
     - Estabelece o ambiente necessário para executar o *MPI*,
     - Sincroniza todos os processos na inicialização de uma aplicação.

2. Identificar processos -

   `MPI_COMM_RANK`

   - Características:
     - Identifica o processo dentro de um grupo de processos,
     - Os n processos são enumerados entre 0 e $n-1$.

3. Contar processos -

   `MPI_COMM_SIZE`

   - Características:
     - Fornece o número de processos dentro de um grupo de processos.

4. Enviar mensagens -

   `MPI_SEND`

   - Características:
     - *Send* bloqueante,
     - A rotina retorna após o dado ter sido enviado,
     - Após o retorno libera o *buffer* do sistema e permite acesso ao *buffer* de aplicação.

5. Receber mensagens -

   `MPI_RECV`

   - Características:
     - *Receive* bloqueante,
     - A rotina retorna após o dado ter sido recebido e armazenado,
     - Após o retorno, libera o *buffer* do sistema.

6. Finalizar processos -

   `MPI_FINALIZE`

   - Características:
     - Finaliza o processo para o **MPI**,
     - Última rotina a ser executada,
     - Sincroniza todos os processos na finalização de uma aplicação.

Apêndice D

Otimizador-AG sem UD - Simulações

Este apêndice mostra os comportamentos dos autovalores e das sensibilidades dos autovalores, referentes as simulações apresentadas na seção (6.3), ao longo do *ciclo de busca* para o otimizador-AG sem unidade de decisão.

As funções de sensibilidade dos autovalores da Figura (D.1) elucidam a natureza oscilatória da função de custo correspondente a Tarefa 2, Figura (6.2), pois verifica-se que as suas sensibilidades ficam oscilando abaixo da maior sensibilidade a cada geração. Para a Tarefa 3, Figura (D.2), as frequências de oscilação são muito pequenas, e as oscilações ocorrem em torno da população 200, um comportamento com pequenas oscilações que persiste até aproximadamente a população 210 da função de custo, Figura (6.2), correspondente; a partir deste ponto a função de *fitness* assume um comportamento decrescente. Para o processo 1, Figura (D.3), as oscilações praticamente não existem e as suas amplitudes são mínimas, assumindo um comportamento quase que totalmente decrescente, o que explica o comportamento também decrescente da função de *fitness* correspondente para esta tarefa, Figura (6.2).

As Figuras (D.4-D.12) mostram os comportamentos dos autovalores para as três tarefas. As maiores frequências de oscilação ocorrem para o processo 2, Figuras (D.4-D.6); como consequência, tem-se que a sensibilidade dos autovalores e a sua respectiva função de *fitness*, Figura (6.2), também possui as maiores oscilações entre os três processos. Para o processo 3, Figuras (D.7-D.9), as frequências de oscilação já são bem menores e como consequência a sua função de *fitness* possui uma frequência de oscilação bem menor, Fi-

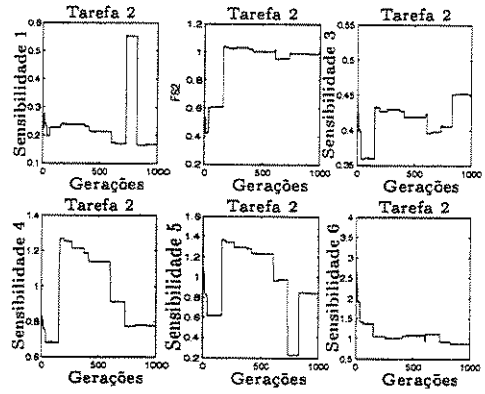

Figura D.1: Sensibilidade dos autovalores da Tarefa-2 *versus* Gerações.

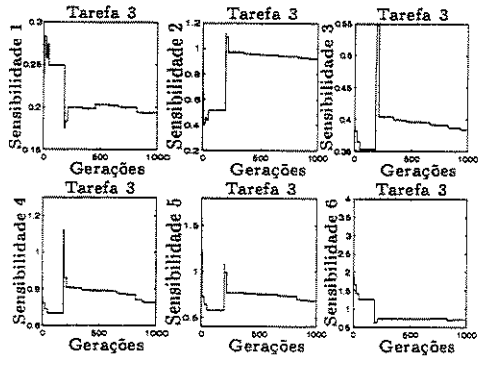

Figura D.2: Sensibilidade dos autovalores da Tarefa-3 *versus* Gerações.

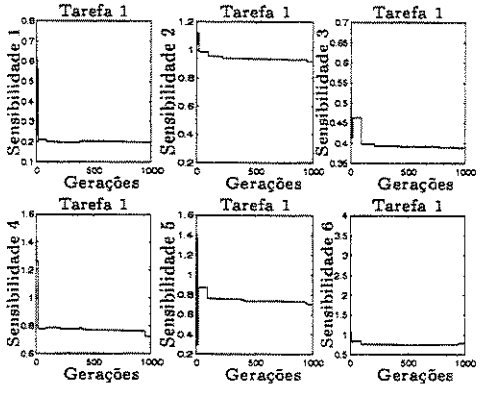

Figura D.3: Sensibilidade dos autovalores da Tarefa-1 *versus* Gerações.

gura (6.2). Já os autovalores da Tarefa 1, Figuras (D.10-D.12) praticamente não apresentam oscilações, mantendo-se praticamente constantes durante a geração de novas populações; para os autovalores 3 e 4 verifica-se mudanças bruscas nas populações iniciais e finais. Observa-se, através de variações dos autovalores que a Tarefa 1 chegou próximo a sua melhor população no início do processo e pequenas melhorias ocorrem para as populações subsequentes; isto pode ser verificado comparando-se seus gráficos com o gráfico da Figura (6.2). Uma conclusão similar pode ser obtida para os processos 2 e 3, através das Figuras (D.4-D.6) e (D.7-D.9), quando são comparadas com a Figura (6.2).

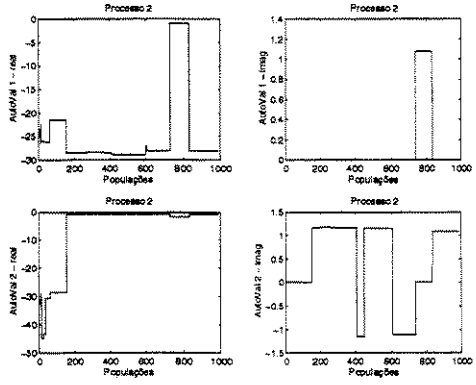

Figura D.4: Autovalores 1 e 2 da Tarefa 2 *versus* Populações.

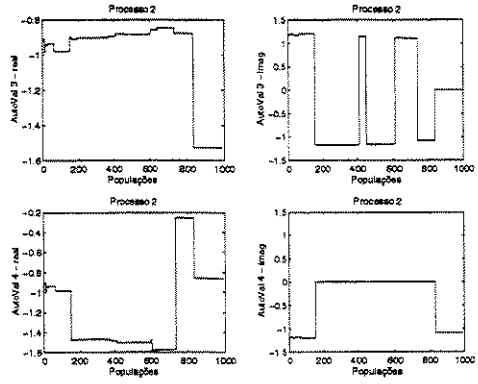

Figura D.5: Autovalores 3 e 4 da Tarefa 2 *versus* Populações.

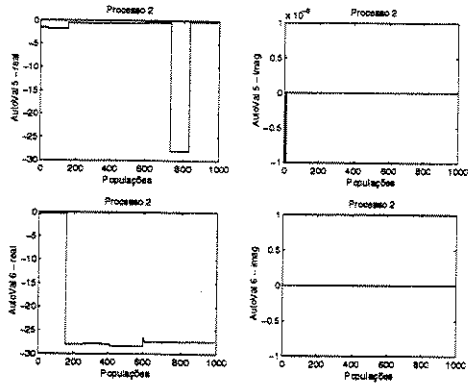

Figura D.6: Autovalores 5 e 6 da Tarefa 2 *versus* Populações.

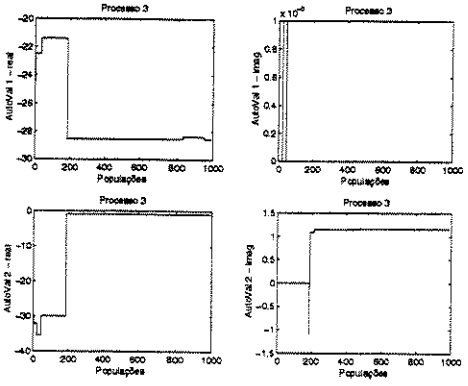

Figura D.7: Autovalores 1 e 2 da Tarefa 3 *versus* Populações.

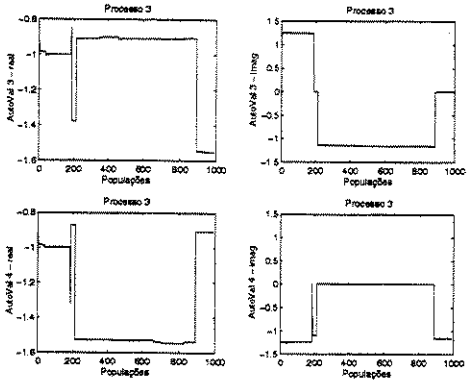

Figura D.8: Autovalores 3 e 4 da Tarefa 3 *versus* Populações.

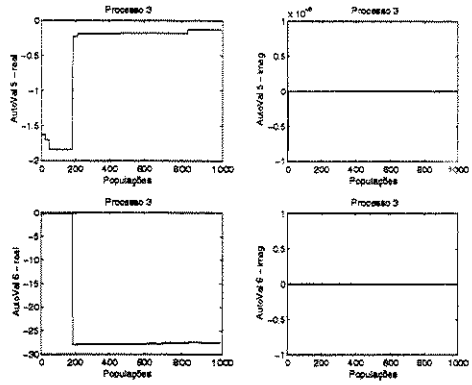

Figura D.9: Autovalores 5 e 6 da Tarefa 3 *versus* Populações.

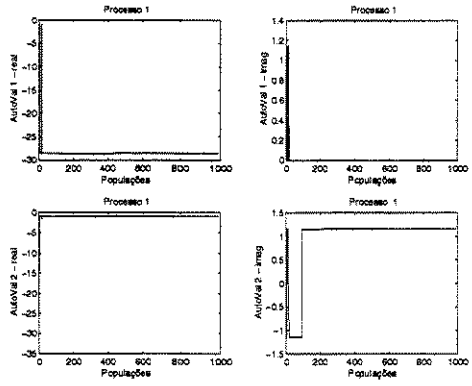

Figura D.10: Autovalores 1 e 2 da Tarefa 1 *versus* Populações.

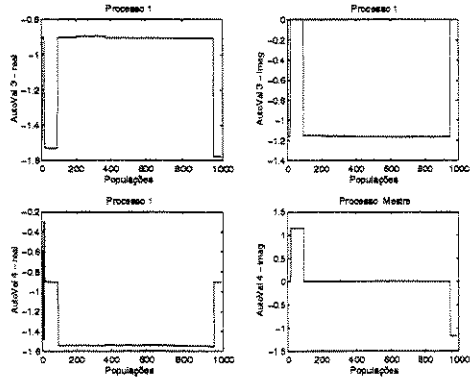

Figura D.11: Autovalores 3 e 4 da Tarefa 1 *versus* Populações.

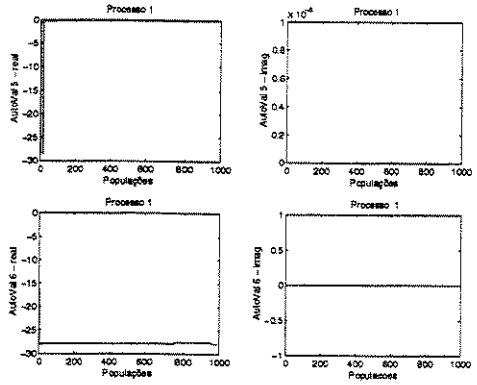

Figura D.12: Autovalores 5 e 6 da Tarefa 1 *versus* Populações.

Apêndice E

Tempos de Processamento sem UD

Este apêndice apresenta medidas de tempo dos *ciclos de buscas*, sem unidade de decisão, para oito tarefas. As simulações foram realizadas em uma rede de estações de trabalho *SUN*, dados sobre as características da rede são encontrados no Apêndice (B). O programa foi instrumentado com rotinas da biblioteca MPE, (Gropp *et al.*, 1994). A finalidade deste apêndice é mostrar como as ferramentas para instrumentação de tempo podem ser utilizadas para melhorar o desempenho do processamento paralelo.

A Tabela (E.1) apresenta medidas de tempo para as oito tarefas. A referida tabela está relacionada com o diagrama de tempo, Figura (E.1). Observando-se os tempos gastos na comunicação entre processos e o tempo de computação de cada processo, verifica-se que o tempo de comunicação é bem menor que o tempo gasto na busca (tempo de computação); logo, o processamento paralelo pode ser caracterizado como de granularidade grossa.

As simulações foram realizadas para um *ciclo de busca* de 100 iterações. As tarefas *escravo* 1 e 2 apresentam o menor tempo de computação, Tabela (E.1), porque as simulações foram feitas nas máquinas *carme* e *io*, *ultra sparc-5*, que são as máquinas mais rápidas da rede. As tarefas *mestre* e *escravo* 3, 4 e 5 foram realizadas em máquinas com diferentes capacidades de processamento, mas apresentaram um tempo de simulação bastante próximo, isto pode ser explicado com auxílio do diagrama de tempo, Figura (E.1); observa-se que a Tarefa *mestre* e a Tarefa *escravo*-3 foram realizadas na mesma máquina, a *aguia-sparc20*, e o *ciclo de busca* foi efetuado em tempo

compartilhado por estas duas tarefas; isto significa que dentro destes tempos deve ser considerado o *overhead* do sistema; por este motivo o tempo de computação da Tarefa 4, realizada em uma máquina *sparc-4*, está próximo dos tempos de computação das tarefas *mestre* e *escravo*-3. As tarefas *escravo* 4 e 5 foram realizadas por máquinas com potência computacional similar, *sparc-4*; contudo, observa-se um alto tempo de computação para a Tarefa-5; esta discrepância pode estar relacionada com a grande solicitação de uso da máquina *saturno*. As tarefas *escravo* 6 e 7 foram realizadas nas máquinas mais lentas da rede, *sparc-2*, e como estas máquinas são pouco utilizadas, as mesmas apresentam um tempo de computação praticamente igual.

O diagrama de tempo, Figura (E.1), é uma ferramenta útil para avaliar e melhorar o desempenho do processamento paralelo. Através dela verifica-se que o desempenho do processamento paralelo pode ser melhorado, se a máquina *carme* for responsável pela distribuição das tarefas *escravo*, porque esta máquina além de ser mais rápida apresentou melhor desempenho e possibilita o recebimento dos resultados das outras tarefas com *overhead* de comunicação minimizado.

Tarefas	*Tempos*- segundos					Máquinas
	M-E	E-R	Computação	E-E	M-R	
Mestre	5.210	–	82.475	–	171.981	aguia
Escravo-1	–	1.074	23.565	64.436	–	carme
Escravo-2	–	1.132	40.347	47.924	–	io
Escravo-3	–	1.105	79.655	8.820	–	aguia
Escravo-4	–	0.679	80.132	8.507	–	urano
Escravo-5	–	6.009	140.094	0.080	–	saturno
Escravo-6	–	5.322	254.084	0.108	–	plutao
Escravo-7	–	5.507	244.527	0.109	–	marte

Tabela E.1: Processamento paralelo: Tempos de envio, de recebimento de dados e de computação para 8 tarefas. Convenção: M-E → Mestre Envia, E-R → Escravo Recebe, E-E → Escravo Envia e M-R → Mestre Recebe.

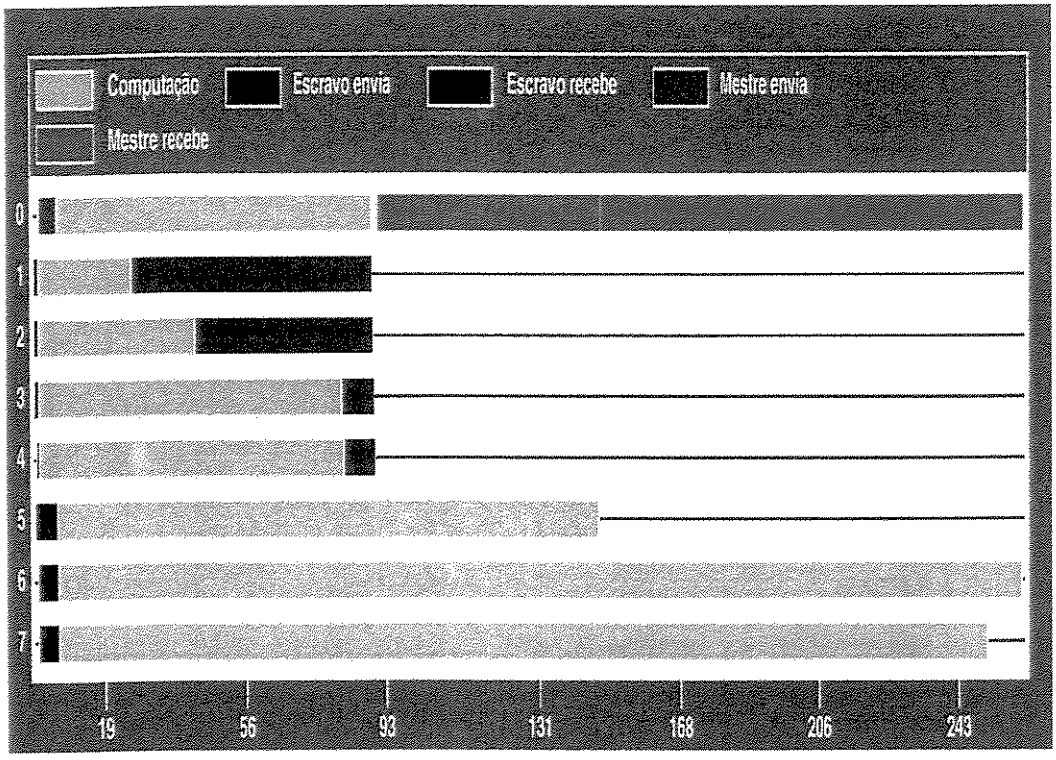

Figura E.1: Comportamento do processamento paralelo para 8 tarefas.

Apêndice F

Otimizador-AG com UD - Simulações

Este apêndice mostra os resultados da simulação sequencial, referente à seção (6.4). Nestas simulações, o otimizador-AG foi guiado por estratégias, fundamentadas na teoria do *schemata*, paradigma do *multiarmed bandit* e time de *fitness*, implementadas na unidade de decisão.

As Figuras (F.1) e (F.2) apresentam os resultados do comportamento da busca e as respostas ao sinal impulso para a simulação sequencial. Como pode ser visto, houve uma melhoria no perfil do melhor indivíduo, Figura (F.1), e da população final, Figura (F.2), para uma busca de 100 iterações. O controlador referente ao melhor indivíduo também apresentou um bom desempenho, Figura (F.2).

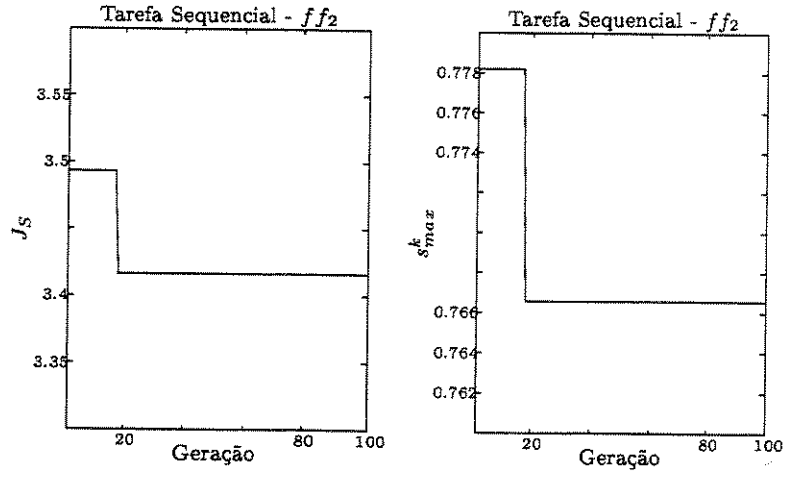

Figura F.1: Tarefa Sequencial - Geração × ff_2 Função de Custo e máximo s^k.

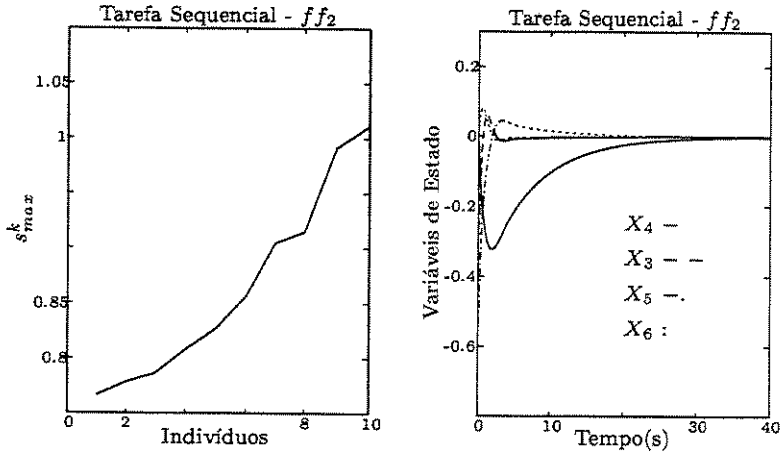

Figura F.2: Tarefa Sequencial - Perfil dos Indivíduos da População Final × ff_2 máximo s^k e resposta ao impulso para o controlador ff_2.

Apêndice G

UD baseada em Regras - Simulações

O principal objetivo destas simulações é verificar os desempenhos das regras R_{idad} e Q_{idad}. A eficiência das regras é verificada para casos em que o otimizador-AG e a *UD* não conseguiram determinar soluções factíveis.

As Figuras (G.1-G.4) apresentam as modificações nos parâmetros da operação *X-over* fornecidas pelas regras e as piores sensibilidades dos autovalores $s_i(Q,R)$ para 4 das oito 8 tarefas. O principal objetivo destes resultados é provar computacionalmente os efeitos das regras propostas para um caso em que não houve convergência e para o qual todas as 8 tarefas conseguiram apresentar soluções factíveis.

De forma comparativa, cada Figura (G.1-G.4), apresenta os resultados quando as regras estão ativadas e não ativadas pela *UD*, ilustrando a evolução das variações dos parâmetros, r_{idad} e q_{idad}, e das sensibilidades dos autovalores para a mesma tarefa.

A Figura (G.1), apresenta quatro curvas (a, b, c e d); a Figura (G.1a) apresenta a variação dos parâmetros dentro de um intervalo e dentro deste intervalo não existem modificações; a Figura (G.1b) apresenta a pior sensibilidade do melhor indivíduo a cada iteração do *ciclo de busca*, que pode ter surgido das variações dos parâmetros apresentados na Figura (G.1a); a Figura (G.1c) apresenta as variações dos parâmetros de acordo com as regras R_{idad} e Q_{idad}; estes parâmetros assumem novos valores todas as vezes que ocorre a operação de *X-over*; a Figura (G.1d) apresenta, de forma similar à Figura (G.1b), a pior sensibilidade correspondente à ação das referidas regras; este padrão de exposição de resultados é válido para as Figuras (G.1)

até (G.4), onde as curvas a) e b) referem-se a não ação das regras e as Figuras c) e d) referem-se à sua atuação. Como pode ser verificado, as regras propostas apresentaram resultados satisfatórios para as 4 tarefas, porque todas as Tarefas *escravo* não tiverem a habilidade de apresentar soluções factíveis sem a ativação das regras R_{idad} e Q_{idad}. Para o caso da Tarefa *mestre*, esta apresentou uma convergência para um número menor de iterações e um melhoria no valor de Δ_k, Figura (G.1).

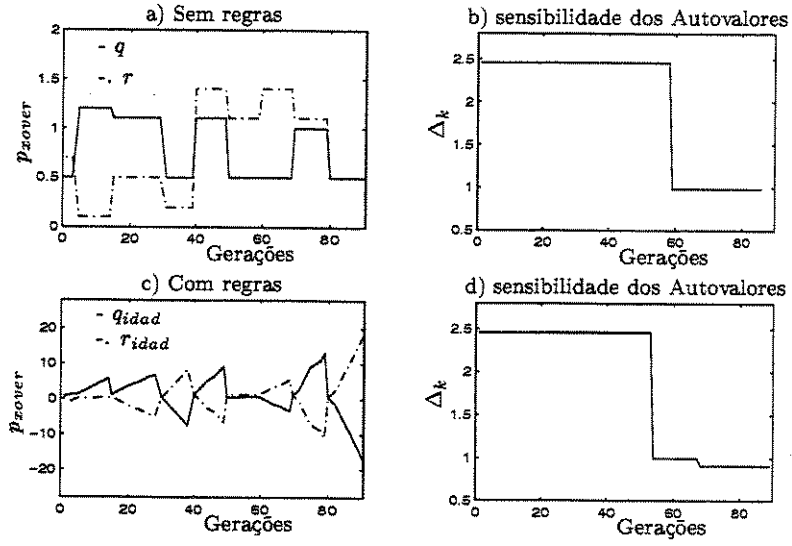

Figura G.1: Tarefa Mestre - modificação nos parâmetros *x-over* e pior sensibilidade dos autovalores *VERSUS* Gerações.

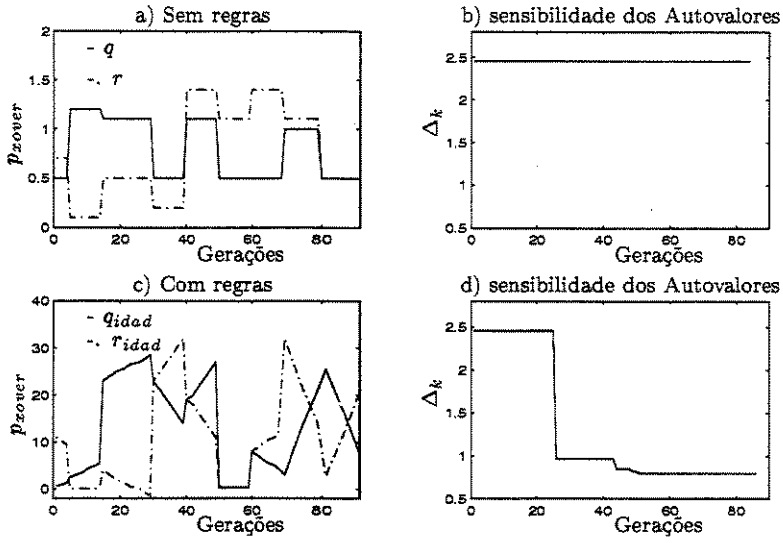

Figura G.2: Tarefa escravo 01 - modificação nos parâmetros *x-over* e pior sensibilidade dos autovalores *VERSUS* Gerações.

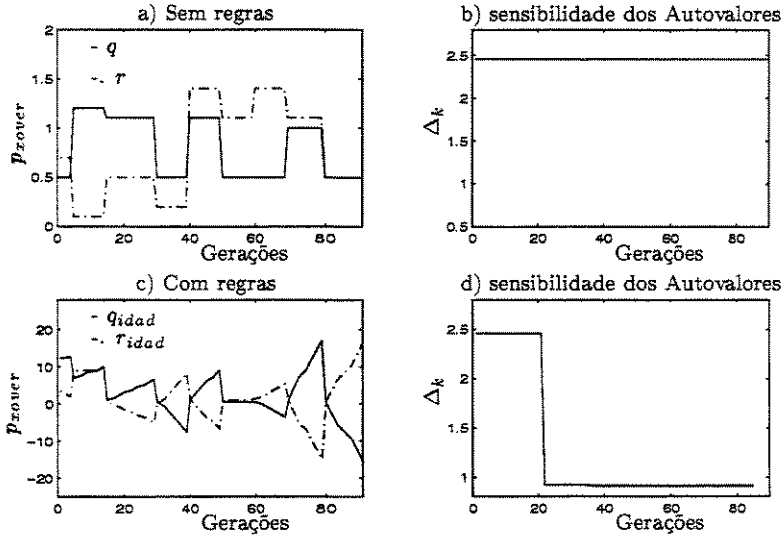

Figura G.3: Tarefa escravo 02 - modificação nos parâmetros *x-over* e pior sensibilidade dos autovalores *VERSUS* Gerações.

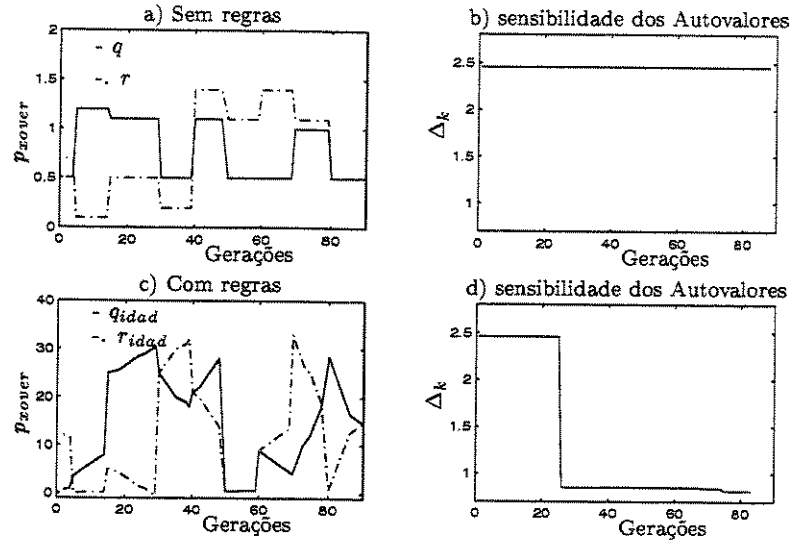

Figura G.4: Tarefa escravo 03 - modificação nos parâmetros *x-over* e pior sensibilidade dos autovalores *VERSUS* Gerações.

O perfil da população pode ser traduzido como um índice de desempenho qualitativo que representa a qualidade da busca do otimizador-AG e da atuação da *UD*. As Figuras (G.5-G.9) apresentam os perfis da população através da pior sensibilidade dos autovalores, Δ_k, e através da função de custo soma das sensibilidades. Comparações são feitas entre as populações inicial e finais quando as regras atuam e quando estas estão desativadas na *UD*. Conclui-se, através de observações nas referidas figuras, que os perfis das populações finais sofrem sensíveis melhorias, quando as regras atuam via *UD*, não só em relação à população inicial, como também em relação à população final sem a ação das regras.

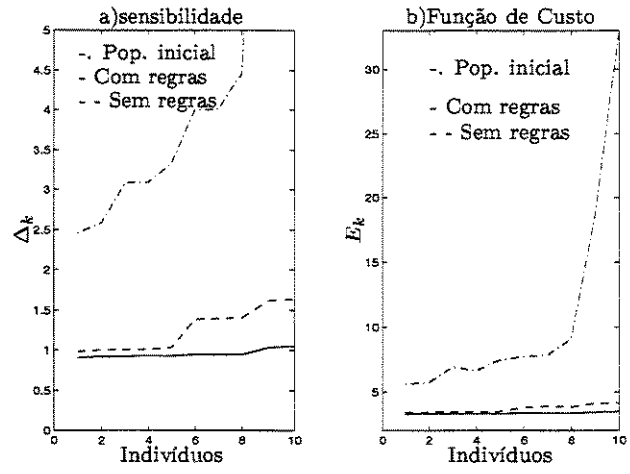

Figura G.5: Perfil da Tarefa Mestre - a) Piores sensibilidades. b)Função de Custo soma das sensibilidades.

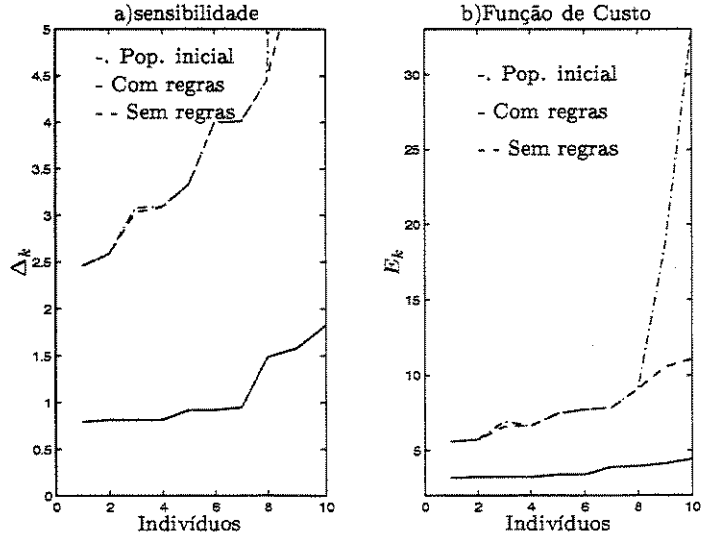

Figura G.6: Perfil da Tarefa Escravo 1 - a) Piores sensibilidades. b)Função de Custo soma das sensibilidades.

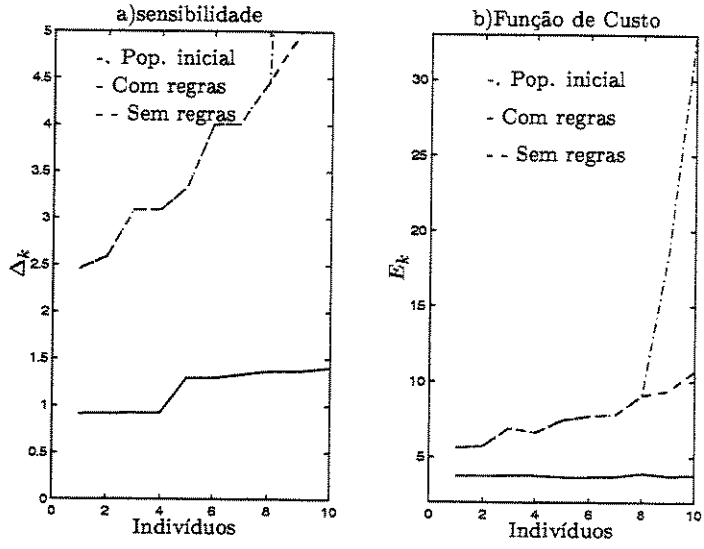

Figura G.7: Perfil da Tarefa Escravo 2 - a) Piores sensibilidades. b)Função de Custo soma das sensibilidades.

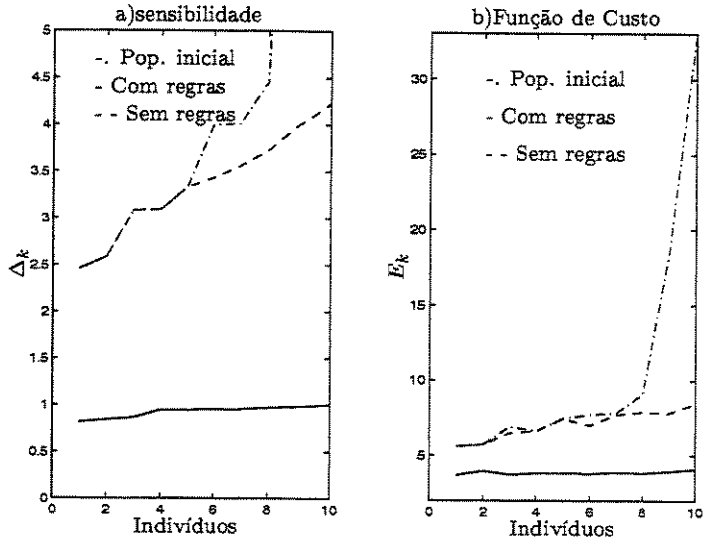

Figura G.8: Perfil da Tarefa Escravo 3 - a) Piores sensibilidades. b)Função de Custo soma das sensibilidades.

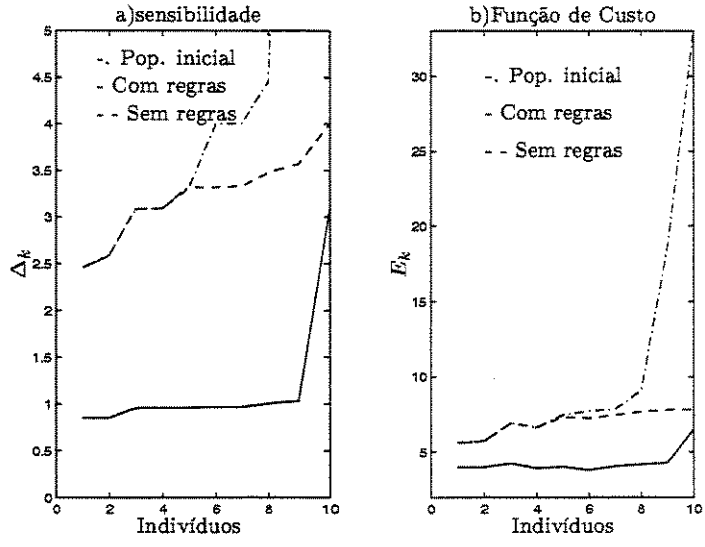

Figura G.9: Perfil da Tarefa Escravo 4 - a) Piores sensibilidades. b)Função de Custo soma das sensibilidades.

Dois tipos de simulações foram realizados para verificar os desempenhos dos controladores, considerando as respostas ao sinal impulso. O primeiro, considera uma família de controladores proveniente de uma mesma tarefa. O segundo tipo, leva em consideração os controladores que apresentaram a pior menor sensibilidade entre todos os controladores provenientes das 8 tarefas distribuídas; a equação (G.1) expressa este tipo de controlador. Outro tipo de simulação, similar ao primeiro tipo, foi realizado considerando uma família de controladores que apresentam pequenas diferenças entre as piores sensibilidades e pequenas diferenças entre os autovalores correspondentes:

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

onde n é a ordem do sistema dinâmico, m é o tamanho da população permanente e l é o número de tarefas distribuídas. Para a simulação do tipo 1 (um), l é a quantidade de controladores que apresentou o menor Δ_k.

As Figuras (G.10-G.11) apresentam os resultados associados com a simulação do tipo 1 e a família de controladores é proveniente da Tarefa *escravo* 3.

Estes resultados são comparados com a resposta ao impulso do controlador de referência e verifica-se que os controladores, com exceção do controlador 3, Figura (G.10d), não apresentaram um desempenho satisfatório para a variável de estado X_4; para mais tentativas um melhor controlador 3 poderia ter sido obtido.

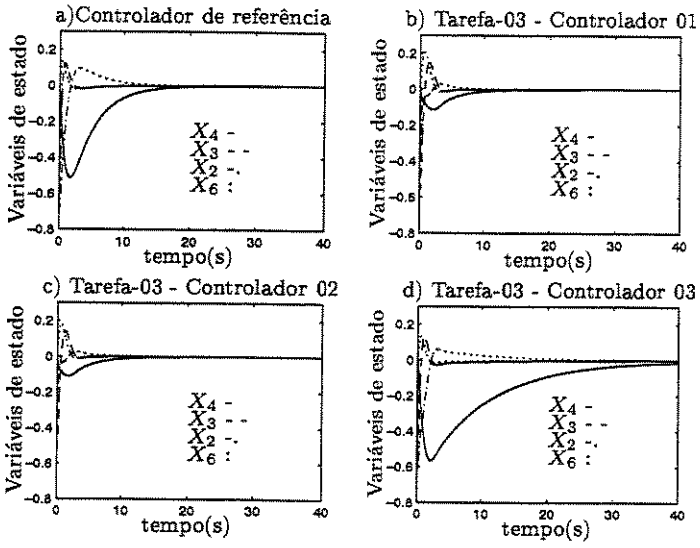

Figura G.10: Respostas ao sinal impulso - Tarefa 03 - Controladores 01-02.

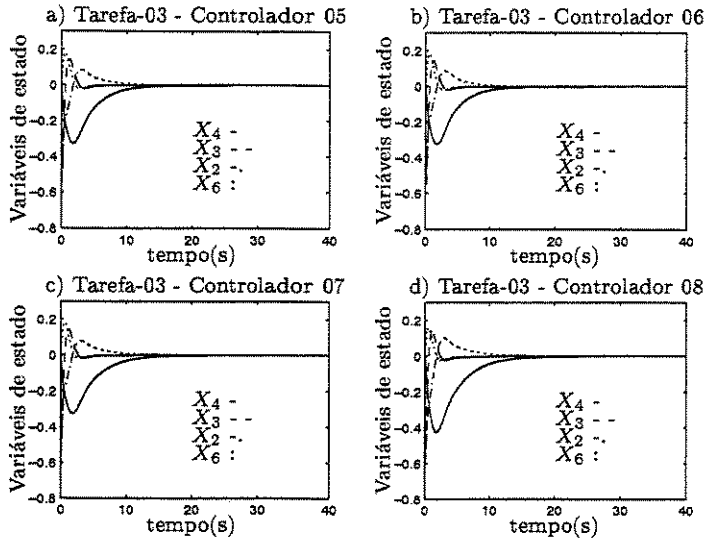

Figura G.11: Respostas ao sinal impulso - Tarefa 03 - Controladores 05-08.

A Tarefa 8 apresenta uma família de controladores com características particulares. Alguns dos controladores factíveis possuem as piores sensibilidades muito próximas, mas seus autovalores não estão tão próximos. As Figuras (G.12-G.13) apresentam os resultados de uma simulação para controladores que gozam destas características. Os controladores 2 até 6 possuem valores de Δ_k muito próximos: 0.823, 0.831, 0.833, 0.837 e 0.839, respectivamente, mas as respostas ao impulso não apresentam um alto grau de similaridade, porque as suas sensibilidades não correspondem ao mesmos autovalores.

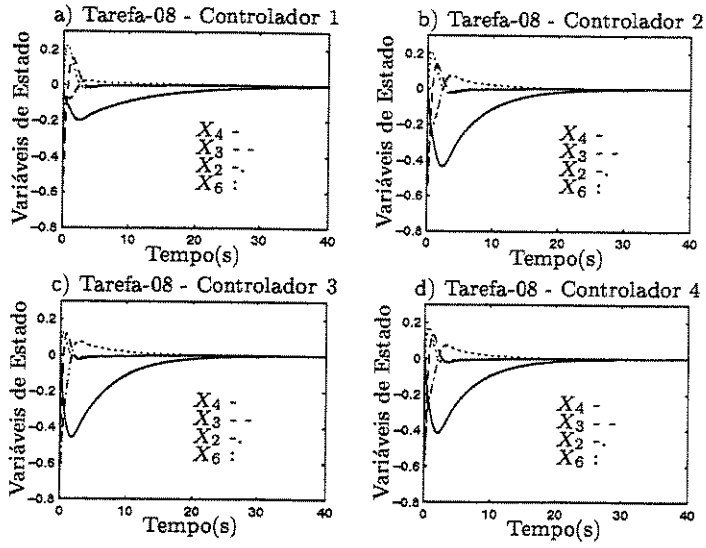

Figura G.12: Respostas ao sinal impulso - Tarefa escravo 8 - Controladores 1-4.

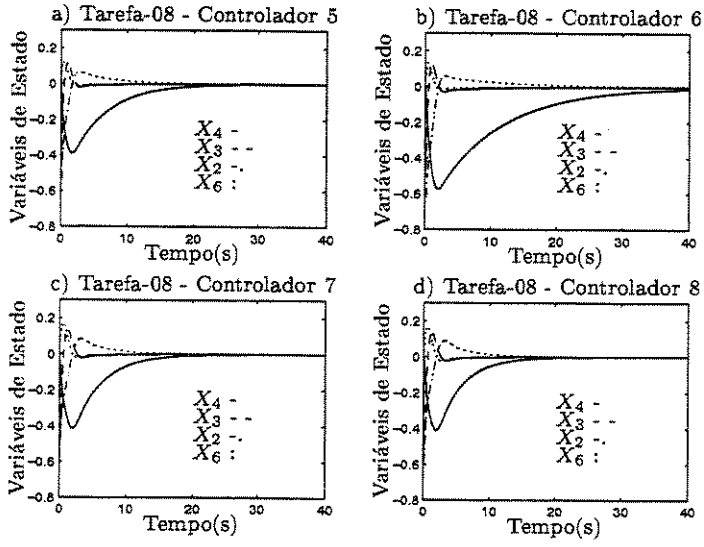

Figura G.13: Respostas ao sinal impulso - Tarefa escravo 8 - Controladores 5-8.

Apêndice H

Ajuste *On-line* - Simulações

Este apêndice complementa os resultados de ajuste de parâmetros *on-line* da operação *X-over*, apresentados na seção (6.6). Analisa-se os comportamentos das ações do ajuste para os três tipos de regras. Os desempenhos das regras são estudados através de simulações exaustivas e das repostas ao sinal impulso, aplicado na entrada do modelo do sistema dinâmico, para uma família de controladores.

H.1 As Acões de Ajuste - Comportamentos

A Figura (H.1) apresenta os resultados para a regra quase-dinâmica. Como pode ser observado, a busca não consegue determinar soluções factíveis para a Tarefa *mestre*, Figura (H.1c), bem como não apresenta controladores factíveis para as Tarefas *escravo*, contudo ela melhorou o perfil da população final, Figura (H.1d).

A regra adaptativa e seus passos de variações, $\Delta q(t)$ e $\Delta r(t)$, são os elementos que influenciam diretamente a formação de novos indivíduos-QR. Esta regra provocou melhorias na busca para $\Delta q(t) \neq 0$ e $\Delta r(t) \neq 0$, Figura (H.4). As curvas das Figuras (H.4a) e (H.4b) estão relacionadas com deslocamentos de alguns indivíduos-QR de alguma região para novos quadrantes e no começo de cada ponto de troca pode existir a oportunidade de realizar uma exploração local do quadrante ou mesmo de permanecer na região até a ocorrência de um novo ponto de troca. Depois da ocorrência de um certo número de gerações, após 100 iterações do *ciclo de busca*, os parâmetros $q(t)_{idad}$ e $r(t)_{idad}$ aceitam pequenas variações aleatórias, isto é, esta política

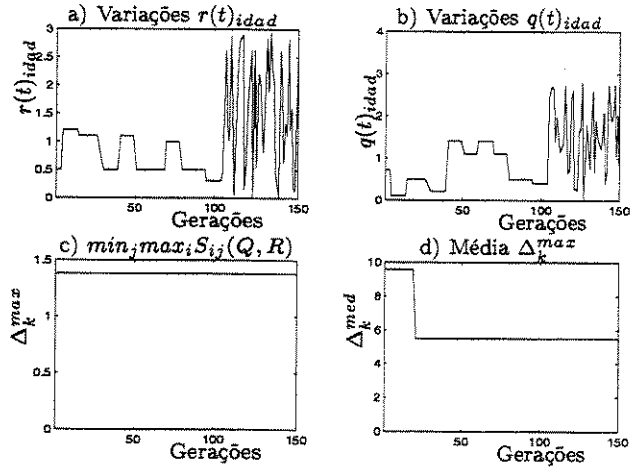

Figura H.1: Regra quase-dinâmica - Variações nos parâmetros da operação *X-over*, máxima sensibilidade e sensibilidade média *versus* gerações.

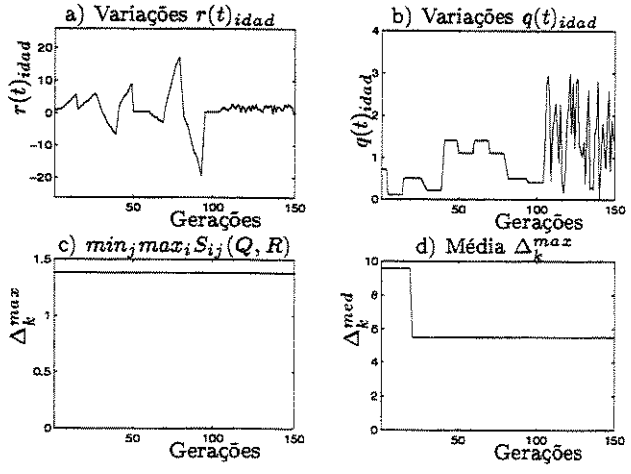

Figura H.2: Regra determinística-direita para $\Delta q(t) = 0$ e $\Delta r(t) \neq 0$ - Variações nos parâmetros da operação *X-over*, máxima sensibilidade e sensibilidade média *versus* gerações.

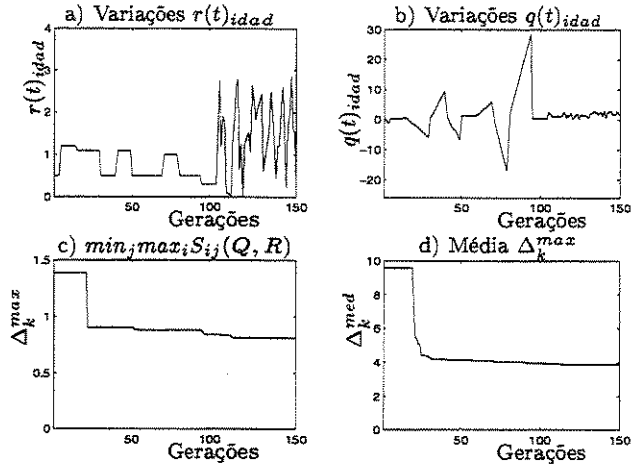

Figura H.3: Regra determinística-direita para $\Delta q(t) \neq 0$ e $\Delta r(t) = 0$ - Variações nos parâmetros da operação *X-over*, máxima sensibilidade e sensibilidade média *versus* gerações.

permite intensificar a busca em uma região limitada do espaço de solução. A Figura (H.4c) mostra que aproximadamente na metade do *ciclo de busca*, o primeiro controlador factível foi obtido e para o resto da busca todos os indivíduos são controladores factíveis, Figura (6.10b). A média, Δ_k^{max}, Figura (H.4d) foi reduzida antes do primeiro quarto do *ciclo de busca* e a mesma redução também ocorreu para a regra quase-dinâmica, Figura (H.1d) e as melhorias mais significantes ocorreram depois da centésima geração. Para $\Delta q(t) = 0$ ou $\Delta r(t) = 0$, Figuras (H.5-H.6), as regras não tiveram a habilidade de determinar controladores factíveis.

As Figuras (H.7-H.9) apresentam o comportamento e os efeitos da regra dinâmica determinística-direita. Uma análise similar à que foi realizada para a regra adaptativa pode ser feita para o desempenho desta regra. Como pode ser visto, esta regra apresentou controladores factíveis para $\Delta q(t) \neq 0$ e $\Delta r(t) \neq 0$, Figura (H.7c), $\Delta q(t) \neq 0$ e $\Delta r(t) = 0$, e para a Figura (H.9c), respectivamente; mas o segundo caso apresentou um controlador factível para menos iterações do *ciclo de busca*. Para o caso, $\Delta q(t) = 0$ e $\Delta r(t) \neq 0$, Figura (H.8), a regra não apresentou controladores factíveis e não promoveu melhorias na média Δ_k^{med}.

Os resultados da regra determinística-esquerda podem ser visualizados

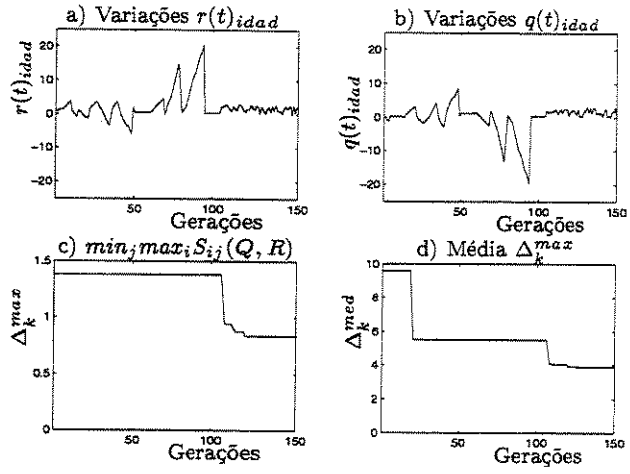

Figura H.4: Regra adaptativa $\Delta q(t) \neq 0$ e $\Delta r(t) \neq 0$ - Variações nos parâmetros da operação *X-over*, máxima sensibilidade e sensibilidade média *versus* gerações.

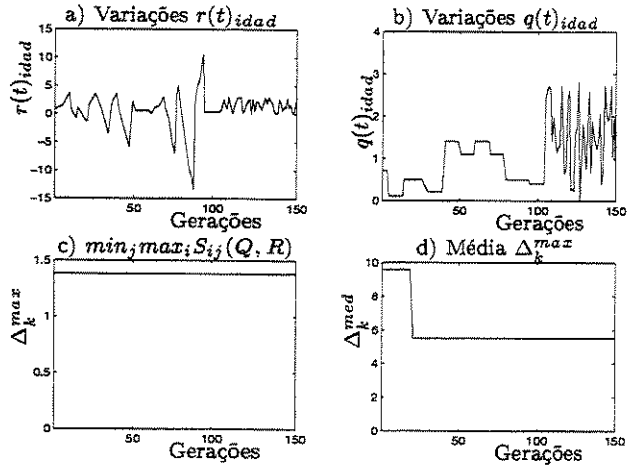

Figura H.5: Regra adaptativa para $\Delta q(t) = 0$ e $\Delta r(t) \neq 0$ - Variações nos parâmetros da operação *X-over*, máxima sensibilidade e sensibilidade média *versus* gerações.

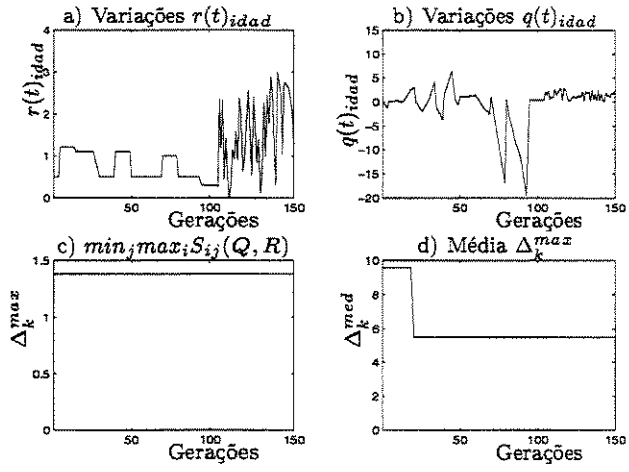

Figura H.6: Regra adaptativa para $\Delta q(t) \neq 0$ e $\Delta r(t) = 0$ - Variações nos parâmetros da operação *X-over*, máxima sensibilidade e sensibilidade média *versus* gerações.

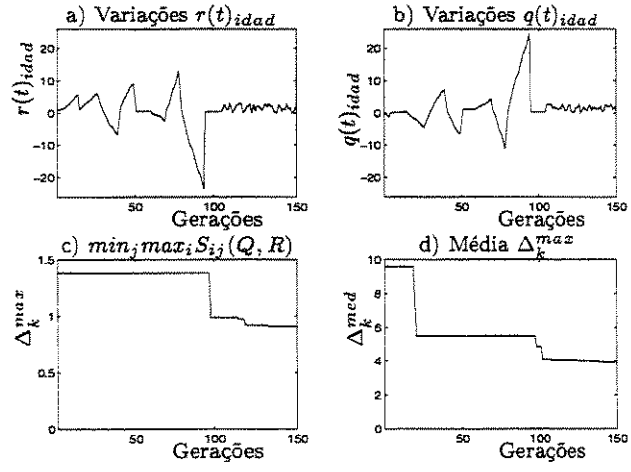

Figura H.7: Regra determinística-direita para $\Delta q(t) \neq 0$ e $\Delta r(t) \neq 0$ - Variações nos parâmetros da operação *X-over*, máxima sensibilidade e sensibilidade média *versus* gerações.

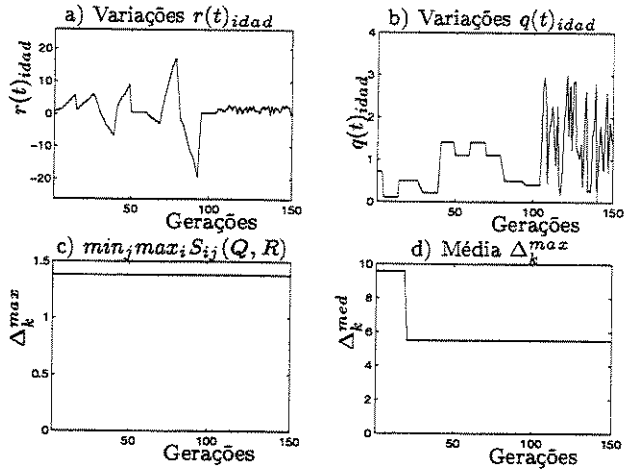

Figura H.8: Regra determinística-direita para $\Delta q(t) = 0$ e $\Delta r(t) \neq 0$ - Variações nos parâmetros da operação *X-over*, máxima sensibilidade e sensibilidade média *versus* gerações.

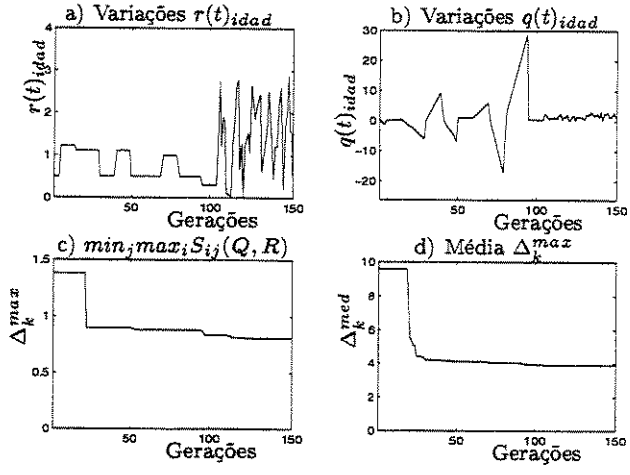

Figura H.9: Regra determinística-direita para $\Delta q(t) \neq 0$ e $\Delta r(t) = 0$ - Variações nos parâmetros da operação *X-over*, máxima sensibilidade e sensibilidade média *versus* gerações.

nas Figuras (H.10-H.12). Apenas para o caso $\Delta q(t) \neq 0$ e $\Delta r(t) = 0$, Figura (H.12), a regra apresentou melhorias na busca e a população permanente final é composta de seis controladores factíveis, Figura (6.10d). O primeiro controlador factível foi obtido em torno da 50ª geração, Figura (H.12c). Os outros dois casos, Figuras (H.10) e (H.11), não apresentaram melhorias durante as iterações do *ciclo de busca*, além das melhorias promovidas no perfil da população final e na média Δ_k^{med}.

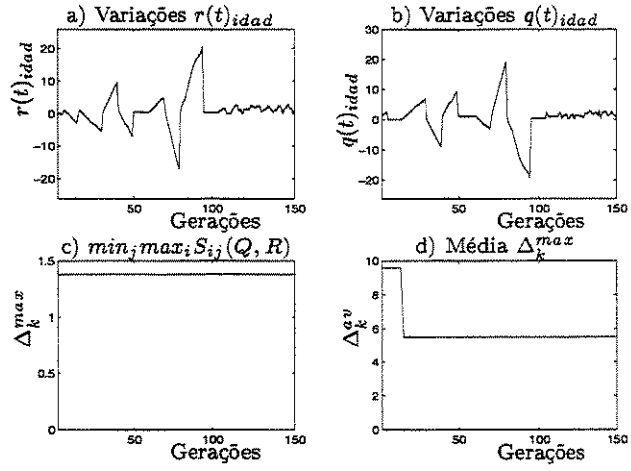

Figura H.10: Regra determinística-esquerda para $\Delta q(t) \neq 0$ e $\Delta r(t) \neq 0$ - Variações nos parâmetros da operação *X-over* variações, máxima sensibilidade e sensibilidade média *versus* gerações.

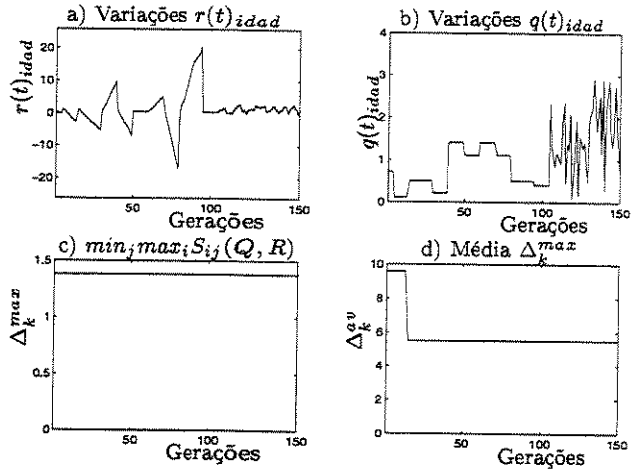

Figura H.11: Regra determinística-esquerda para $\Delta q(t) = 0$ e $\Delta r(t) \neq 0$ - Variações nos parâmetros da operação *X-over* variações, máxima sensibilidade e sensibilidade média *versus* gerações.

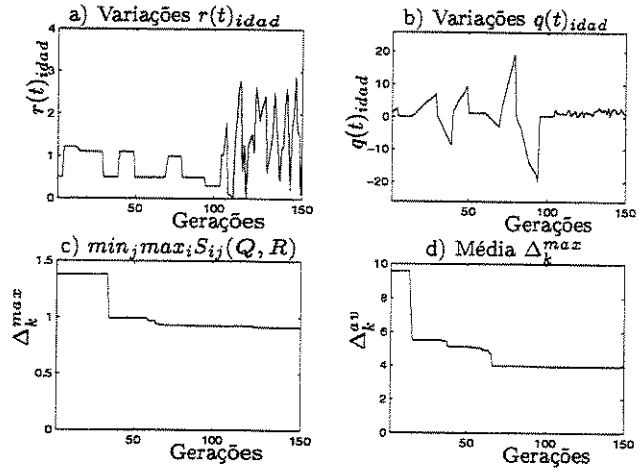

Figura H.12: Regra determinística-esquerda para $\Delta q(t) \neq 0$ e $\Delta r(t) = 0$ - Variações nos parâmetros da operação *X-over*, máxima sensibilidade e sensibilidade média *versus* gerações.

H.2 Os Desempenhos das Regras - Simulações Exaustivas

As simulações exaustivas foram realizadas para 10 diferentes casos. Estes não apresentam soluções factíveis quando os parâmetros da operação *X-over* não sofrem ajuste dinâmico durante as iterações do *ciclo de busca.*

O desempenho da regra adaptativa pode ser observado na Tabela (H.1). Esta Tabela apresenta uma visão global de cada caso, mostrando a pior sensibilidade do melhor indivíduo-QR, Δ_i^{max}, e a geração em que esta foi atingida para cada tarefa. Os casos sem convergência são representados apenas pelo melhor valor de Δ_i^{max} obtido por todas as iterações do *ciclo de busca.* Esta regra melhorou a busca no sentido de que cada um dos dez casos apresentou no mínimo uma solução e no mínimo uma destas soluções foi obtida antes da 100ª geração, isto é, durante iterações em que as regras estavam totalmente ativadas e mesmo para as situações em que estas regras estavam desativadas, depois da geração 105ª, estas contribuíram para a obtenção de soluções factíveis.

A regra determinística-direita, Tabela (H.2), e a regra determinística-esquerda, Tabela (H.3), tiveram a habilidade de promover melhorias substanciais na busca realizada pelo otimizador-AG. Tal qual os resultados obtidos através da regra adaptativa, no mínimo um processo de cada caso apresentou uma solução abaixo da 100ª geração.

Analisando as informações contidas nas Tabelas (H.1-H.3), verifica-se que para cada 80 simulações, as seguintes triplas de estatísticas podem ser obtidas para cada regra, onde os elementos de uma tripla representam a quantidade de convergências abaixo de 100 gerações, acima de 100 gerações e a quantidade de processos sem convergência, respectivamente, bem como seus respectivos percentuais. Os valores atribuídos aos elementos de cada tripla são: (40-50%, 13-16.25%, 27-33.75%) para a regra adaptativa, (55-68.75%, 12-15.00%,13-12.25%) para a regra determinística-direita, (44- 55%, 20-25%,16-20%) para a regra determinística-esquerda.

Uma análise comparativa em termos das quantidades de soluções factíveis e das quantidades máximas e mínimas de buscas realizadas pelos processos de cada caso é feita com as informações apresentadas nas Tabelas (H.4-H.6).

Casos	Valores observados	Processos							
		Mestre	Escravos						
			01	02	03	04	05	06	07
1	Δ_i^{max}	0.94	1.38	1.38	1.38	0.90	0.90	1.38	0.87
	Geração	108	-	-	-	110	68	-	2
2	Δ_i^{max}	0.97	0.88	0.88	2.92	0.93	0.95	2.92	0.87
	Geração	94	76	88	-	83	82	-	79
3	Δ_i^{max}	0.99	1.62	0.99	0.93	0.93	0.93	0.84	0.97
	Geração	3	-	3	93	51	3	29	143
4	Δ_i^{max}	0.96	1.57	1.57	0.99	0.90	0.91	0.99	0.99
	Geração	29	-	-	74	34	90	50	71
5	Δ_i^{max}	2.52	0.91	0.98	2.52	0.90	0.96	0.99	0.96
	Geração	-	78	101	-	87	102	87	118
6	Δ_i^{max}	1.52	1.52	0.99	1.52	1.52	1.52	1.52	1.52
	Geração	-	-	25	-	-	-	-	-
7	Δ_i^{max}	1.53	0.95	0.88	1.53	1.53	0.98	1.53	0.99
	Geração	-	88	183	-	-	121	-	60
8	Δ_i^{max}	0.95	0.93	0.85	0.90	0.90	2.08	2.08	0.89
	Geração	39	28	83	76	83	-	-	150
9	Δ_i^{max}	0.94	1.18	1.18	1.18	0.99	0.98	0.98	0.99
	Geração	4	-	-	-	8	64	24	5
10	Δ_i^{max}	0.96	0.99	0.94	0.91	0.89	0.95	0.95	0.90
	Geração	79	87	116	86	53	188	119	125

Tabela H.1: Resultados de exaustivas simulações dos ajustes da regra adaptativa para $\Delta_q(t) \neq 0$ e $\Delta_r(t) \neq 0$.

Casos	Valores Observados	Processos							
		Mestre	Escravos						
			01	02	03	04	05	06	07
1	Δ_i^{max}	0.99	0.92	1.38	0.99	0.95	0.93	0.94	0.87
	Geração	97	47	-	49	28	134	17	3
2	Δ_i^{max}	0.94	0.79	0.91	0.89	0.85	0.93	0.87	0.79
	Geração	125	80	73	75	94	91	90	79
3	Δ_i^{max}	0.99	0.98	0.99	0.91	0.96	0.99	0.94	0.99
	Geração	3	25	3	76	53	3	163	89
4	Δ_i^{max}	0.90	0.99	0.93	0.99	0.97	1.57	0.99	0.97
	Geração	19	75	30	54	30	-	144	195
5	Δ_i^{max}	2.52	0.96	0.96	2.52	2.52	0.88	0.82	0.97
	Geração	-	78	28	-	-	48	93	58
6	Δ_i^{max}	0.98	1.52	0.99	1.52	1.52	1.52	1.52	0.98
	Geração	42	-	19	-	-	-	-	44
7	Δ_i^{max}	0.99	1.53	0.83	0.91	0.97	0.97	0.99	1.53
	Geração	137	-	133	98	72	185	141	-
8	Δ_i^{max}	0.99	0.91	0.97	0.98	0.94	0.93	0.99	2.08
	Geração	56	27	59	189	28	31	42	-
9	Δ_i^{max}	0.94	0.99	0.96	0.99	0.99	0.98	0.95	0.99
	Geração	4	50	54	107	7	24	22	4
10	Δ_i^{max}	0.92	0.93	0.93	0.88	0.97	0.90	0.89	0.94
	Geração	48	50	55	28	115	40	72	56

Tabela H.2: Resultados de exaustivas simulações dos ajustes da regra determinística-direita para $\Delta_q(t) \neq 0$ e $\Delta_r(t) \neq 0$.

Casos	Valores Observados	Processos							
		Mestre	Escravos						
			01	02	03	04	05	06	07
1	Δ_i^{max}	1.38	0.94	1.38	1.38	0.97	1.38	0.96	1.38
	Geração	-	94	-	-	59	-	91	-
2	Δ_i^{max}	0.78	0.94	0.98	0.84	0.94	0.94	0.78	0.94
	Geração	88	64	94	88	94	90	88	101
3	Δ_i^{max}	0.95	0.95	0.94	0.93	0.90	0.97	0.96	0.99
	Geração	37	107	67	136	39	83	80	35
4	Δ_i^{max}	0.99	0.92	0.98	0.93	0.99	0.95	0.99	0.99
	Geração	119	36	125	84	37	52	50	35
5	Δ_i^{max}	0.93	0.86	0.94	0.93	0.84	0.86	0.95	0.87
	Geração	111	180	86	143	113	143	83	95
6	Δ_i^{max}	1.52	0.93	1.52	1.52	0.90	1.52	0.96	1.52
	Geração	-	126	-	-	150	-	62	-
7	Δ_i^{max}	1.53	0.99	0.92	1.53	0.92	0.95	0.85	0.99
	Geração	-	90	173	-	58	35	189	52
8	Δ_i^{max}	0.90	0.90	0.92	2.08	2.08	0.99	0.96	2.08
	Geração	55	181	87	-	-	186	148	-
9	Δ_i^{max}	0.97	0.97	0.99	0.94	0.83	0.95	1.18	0.95
	Geração	89	35	85	58	83	60	-	105
10	Δ_i^{max}	0.93	0.89	0.92	0.89	0.96	0.89	0.99	0.95
	Geração	92	94	130	39	186	39	37	88

Tabela H.3: Resultados de exaustivas simulações dos ajustes da regra determinística-esquerda para $\Delta_q(t) \neq 0$ e $\Delta_r(t) \neq 0$.

Caso	Quantidade de Buscas		Sucesso
	Mínima	Máxima	Quantidade
1	2	110	4
2	76	94	6
3	3	143	7
4	29	90	6
5	78	118	6
6	25	25	1
7	60	183	4
8	28	150	6
9	4	64	5
10	53	125	8

Tabela H.4: Resultado de desempenho da regra adaptativa.

As informações contidas nas Tabelas (H.4-H.6) foram obtidas dos resultados apresentados nas Tabelas (H.1-H.3). Comparando os desempenhos das três regras, Tabelas (H.4-H.6), verifica-se que a regra adaptativa apresentou a primeira solução factível e que a maior parte dos seus resultados acontece antes da 100ª geração. As outras duas regras apresentaram um maior número de sucessos que a regra adaptativa. Comparando os desempenhos entre as regras determinísticas, verifica-se que a regra à direita apresentou um maior número de resultados bem sucedidos do que a regra à esquerda, bem como a factibilidade foi atingida para um número menor de iterações.

Caso	Quantidade de Buscas		Sucesso
	Mínima	Máxima	Quantidade
1	59	94	3
2	64	101	8
3	35	136	8
4	35	125	8
5	83	180	8
6	62	150	3
7	52	189	6
8	55	186	5
9	35	105	7
10	37	186	8

Tabela H.5: Resultado de desempenho da regra determinística-direita.

Caso	Quantidade de Buscas		Sucesso
	Mínima	Máxima	Quantidade
1	3	134	7
2	73	125	8
3	3	163	8
4	19	195	7
5	28	93	5
6	19	42	3
7	72	185	6
8	28	189	7
9	4	107	8
10	28	115	8

Tabela H.6: Resultado de desempenho da regra determinística-esquerda.

H.3 Família de Controladores

A Figura, (H.13), apresenta as respostas ao impulso do sistema dinâmico para uma implementação de quatro membros de um família de controladores que envolve um controlador da população transitória final. Estes controladores foram obtidos do caso 1 através da regra adaptativa, Tabela (H.1). Comparando a resposta do sistema para os controladores 1, Figura (H.13a), e 4, Figura (H.13b) verifica-se que os comportamentos da resposta transitória são muito similares, apesar das sensibilidades não assumirem valores próximos: 0.73 e 0.84 para os controladores 1 e 4, respectivamente. As respostas ao impulso do sistema para os controladores 09, Figuras (H.13c) e 14, Figura (H.13d), mostram um maior amortecimento do que para implementações dos controladores 1 e 4; as sensibilidades dos controladores 9 e 14 são 0.90 e 0.99, respectivamente.

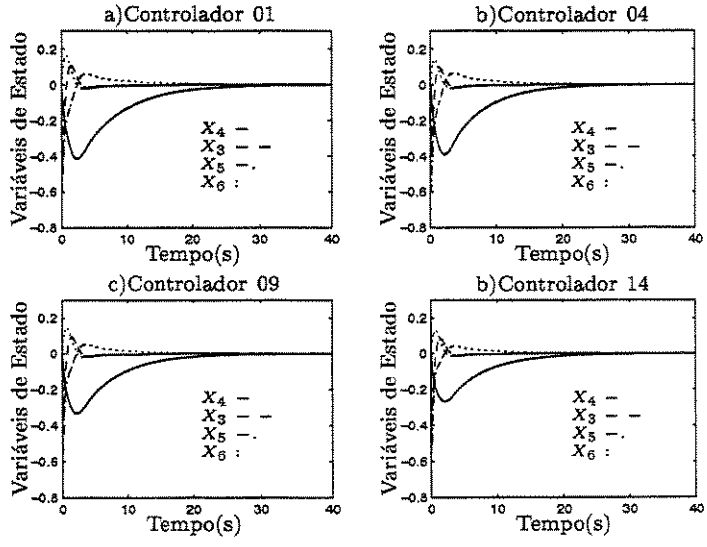

Figura H.13: Caso 1 - Respostas ao impulso - Tarefa 9 - Família de controladores obtidos com a regra adaptativa

Referências Bibliográficas

Adenso, Dias, Fred Glover, Hassan M. Ghaziri, J.L. Gonzalez, Manuel Laguna, Pablo Moscato e Fan T. Seng (1996). *Optimización Heurística y Redes Neuronales.* primera ed.. Editorial Paraninfo S.A.. Magallanes, 25 - Madrid - España.

Alba, Enrique e Carlos Cotta (1997). *The On-line Tutorial on Evolutionary Computation.* Departamento de Lenguages y Ciencias de la Computation - Universidad de Málaga. Málaga- España.

Altenberg, Lee (1995). The Schema Theorem and Price's Theorem. Foundantions of Genetic Algorithms. Edited by L.Darrel Whitley and Michael D. Vose.

Anderson, Brian D. O e John B. Moore (1990). *Optimal Control: Linear Quadratic Methods.* Prentice-Hall, Englewood Cliffs.

Andry Jr., A.N., E.Y. Shapiro e J.C. Chung (1983). Eigenstructure Assignment for Linear Systems. *IEEE Transactions on Aerospace and Electronic Systems* **19**(5), 711–729.

Antsaklis, Panos J. e N. Anthony Michel (1997). *Linear System Theory and Design.* McGraw-Hill.

Athans, M. e H.P Geering (1973). Necessary and sufficient conditions for differentiable nonscalar-valued functions to attain extremes. *IEEE Transactions on Automatic Control* **AC-18**, 132–139.

Athans, Michael e L. Peter Falb (1966). *Optimal Control- An Introduction to the Theory and Its Applications.* McGRAW-Hill Book Company . United States of America.

Barr, R., B. Golden, J. Kelly, M. Resende e W. Stewart (1995). Guidelines for Designing and Reporting on Computacional Experiments with Heuristics Methods. *Journal of Heuristics.*

Bäck, Thomas (1996). *Evolutionary Algorithms in Theory and Practice.* first ed.. Oxford University Press, Inc. 198 Madison Avenue, New York, New York 10016.

Bäck, Thomas, David B. Fogel e Zbigniew Michalewicz (1997). *Handbook of Evolutionary Computation.* first ed.. Institute of Physics Publishing and Oxford University press. Bristol and New York.

Bertsekas, Dimitri P. e John N. Tsitsiklis (1989). *Parallel and Distributed Computation: Numerical Methods.* Prentice-Hall Inc.. Englewood Cliffs, NJ, USA.

Bonissone, Piero, Yu-TO Chen, Kai Goebel e Pratap S. Khedar (1999). Hybrid soft computing systems: Industrial and commercial applications. *Proceedings of the IEEE* **87**(9), 1641–1667.

Bottura, C. P. e J.V. da Fonseca Neto (1999*a*). Parallel Eigenstructure Assignment via LQR Design and Genetic Algorithms. *American Control Conference - ACC99. San Diego, California, USA.* pp. 2295–2296.

Bottura, C. P. e J.V. da 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, C. P. e J.V. da Fonseca Neto (1999*c*). The Schema Theorem and Multiarmed Bandit Paradigm Influences on Eigenstructure Assignment via LQR Designs. 4º *Simpósio Brasileiro de Automação Inteligente - SBAI99. São Paulo - SP - Brasil* pp. 502–506.

Bottura, C. P. e J.V. da Fonseca Neto (1999*d*). X-over Parameter Control for a GA-optimizer dedicated to Eigenstructure Assignment/LQR designs - Part I - Problem Formulation. *Submitted to a Congress.*

Bottura, C. P. e J.V. da Fonseca Neto (1999*e*). X-over Parameter Control for a GA-optimizer dedicated to Eigenstructure Assignment/LQR designs -Part II - Computational Simulations and Performance Analysis. *Submitted to a Congress.*

Bottura, C. P. e J.V. da Fonseca Neto (2000). Rule-based decision making unity for eigenstructure assignment via parallel genetic algorithm and lqr designs. *Aceito na American Control Conference 2000 - ACC2000.*

Bottura, Celso P., Annabell D.R Tamariz, Gilmar Barreto e J.V. da Fonseca Neto (1999*a*). Formas Sequencial e Paralela para Solução da Equação Algébrica de Riccati por um algoritmo de Schur-Modificado. In: *XV Congresso Brasileiro de Engenharia Mecânica.* Águas de Lindóia, SP, Brasil.

Bottura, Celso P., Annabell D.R. Tamariz, Gilmar Barreto e J.V. da Fonseca Neto (1999*b*). Sequential and Parallel Algebraic Riccati Equations via ESST on the Schur Method. In: *Proceedings of* 38^{th} *Conference on Decision & Control.* Phoenix, Arizona, USA. pp. 2739–2740.

Bottura, Celso P. e J.V. da Fonseca Neto (1999*f*). Parallel Eigenstructure Assignment via Linear Quadratic Design and Multiobjective Genetic Algorithm with Double Fitness Functions. In: *XV Congresso Brasileiro de Engenharia Mecânica.* Águas de Lindóia, SP, Brasil.

Bottura, Celso Pascoli (2000). *Anotações para um livro em Controle Multivariável.* Em preparação.

Báran, Benjamín, Eugenius Kaszkurewicz e Amit Bhaya (1996). Paralllel Asynchronous Team Algorithms: Convergence and Performance Analysis. *IEEE Transactions on Parallel and Distributed Systems* **7**(7), 677–688.

Bronson, Richard (1989). *Theory and Problems of Matrix Operations.* McGraw-Hill.

Brown, Donald E., Chistopher L. Huntley e Andrew R. Spillane (1989). A Parallel Genetic Heuristic for the Quadratic Assignment Problem. *Proceedings of* 3^{th} *International Conference on Genetic Algorithms* pp. 406–415.

Bryson, Arthur E. Jr e Yu-Chi Ho (1975). *Applied Optimal Control: Optimization, Estimation and Control.* John Willey & Sons Inc.

Cantú-Paz, E. (1998). A survey of Parallel Genetic Algorithms. *Calculateurs Parallèles, Reseaux et Systems Repartis* **10**(2), 141–171.

Cantú-Paz, Erick (1999). Designing Efficient and Accurate Parallel Genetic Algorithms. Ph.D. Theses. University of Illinois at Urbana-Champaign. Urbana-Illinois.

CENAPAD (1996). *Treinamento: Introdução ao MPI.* Centro Nacional de Processamento de Alto Desempenho. Campinas - SP.

Chang, S.S.L. (1966). General Theory of Optimal Processes. *SIAM Journal of Control* **4**, 46–55.

Chankong, Vira e Yacov Y. Haimes (1983). *Multiobjective Decision Making: Theory and Methodology.* Elsevier Science Publishing Co, Inc.. Amsterdam, Netherlands.

Chen, Chi-Tsong (1984). *Linear System Theory and Design.* Holt, Rinehart and Winston, Inc.

Chipperfield, Andrew e Peter Fleming (1996). Evolutionary Algorithms for Control Engineering. In: *Proceedings 13th Triennial IFAC World Congress - San Francisco - USA.* Vol. 1. pp. 181–186.

Choi, Jae Weon e Young Bong Seo (1999*a*). LQR Synthesis via Eigenstructure Assignment. *IFAC- 14th Triennial World Congress, Beijing, P.R. China* pp. 153–158.

Choi, Jae Weon e Young Bong Seo (1999*b*). LQR Design with Eigenstructure Assignment Capability. *IEEE Transactions on Aerospace and Eletronic Systems* **35**(2), 700–708.

Chyung, D.K. (1967). Optimal systems with multiple cost functionals. *SIAM Journal of Control* **5**, 345–351.

Costa Filho, José Tarcisio (1992). Proposta para Computação Assíncrona Paralela e Distribuída de Estruturas Especiais de Jogos Dinâmicos. Tese de doutorado. Universidade Estadual de Campinas. Campinas - São Paulo.

CSEP (1996). *Computational Science Education Project.* 6 ed.. Sponsored by U.S. Department of Energy. http://csep1.phy.ornl.gov/csep.html.

Dangprasert, Pataya e Vichit Avatchakorn (1996). Genetic Algorithms Based on an Intelligent Controller. *Expert Systems With Applications* **10**(3/4), 465–470.

Darwin, Charles (1859). *The Origin of Species.* 1993 modern library edition ed.. Random House Inc. New York.

Davis, R. e T. Clarke (1995). Parallel Implementation of a Genetic Alghorithm. *Control Eng. Practice* **3**(1), 11–19.

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

Donald, Neil Mac, Elsepeth Minty, Tim Harding e Simon Brown (1998). *Writing Message-Passing Parallel Programs with MPI.* Edinburgh Parallel Computing Center. University of Edinburgh.

Dorato, P., C. Abdallah e V. Cerone (1995). *Linear Quadratic Control.* Prentice-Hall, Englewood Cliffs.

Dorato, Peter (1991). *A survey of Robust Multiobjective Techniques. In: Control of Uncertain Dynamic Systems. Edited by S. P. Battacharryya and L. H. Keel.* CRC Press. Boca Raton, Florida, USA.

Eiben, Á. E., R. Hiterding e Z. Michalewicz (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation* **3**(2), 124–141.

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

Fartes Filho, A. M. (1977). *Análise e Computação de Formas Canônicas para Sistemas Lineares Multivariáveis, Tese de Mestrado.* Universidade Estadual de Campinas - UNICAMP, Campinas, SP, Brasil.

Fletcher, L.R. (1981*a*). On Pole Assignment in Linear Systems with direct feedthrough - I: theoretical considerations . *International Journal of Control* **33**, 739–749.

Fletcher, L.R. (1981*b*). On Pole Assignment in Linear Systems with direct feedthrough - II: computational considerations . *International Journal of Control* **33**, 1147–1154.

Fogel, David B., Toshio Fukuda e Ling Guan (1999). Scanning the issue/technology - special issue on computacional intelligence. *Proceedings of the IEEE* **87**(9), 1415–1421.

Fonseca, Carlos Manoel Mira da (1995). Multiobjective Genetic Algorithms with Application to Control Engineering Problems. Ph.D Thesis. Department of Automatic Control Systems and Engineering. The University of Sheffield. Sheffield, England, UK.

Fonseca, C.M. e P.J. Fleming (1998). Multiobjective Optimization and Multiple Constraint Handling with Evolutionary Algorithms-Part I: A Unified Formulation and Algorithms-Part II: Application Example. *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans* **28**(1), 26–37.

Geist, Al, Adam Beguelin, Jack Dongarra, Weicheng Jiang, Robert Manchek e Vaidy Sunderam (1994). *PVM: Parallel Virtual Machine - A User's Guide and Tutorial for Networked Parallel Computing.* MIT Press. Cambridge - Massachusetts. Third Printing 1996.

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

Gorges-Schleuter, Marina (1989). ASPARAGOS An Asynchronous Parallel Genetic Optimization Strategy. *Proceedings of* 3^{th} *International Conferece on Genetic Algorithms* pp. 421–427.

Graupe, D. (1972). Derivation of Weighting Matrices towards satisfying Eigenvalue requirements. *Int. J. Control* **16**(5), 881–888.

Grefenstette, J.J. (1981). Parallel adaptive algorithms for function optimization. Technical Report CS-81-19. Nashville: Vanderbilt University. Computer Science Department.

Gropp, William, Ewing Lusk e Anthony Skjllum (1994). *USING MPI: Portable Parallel Programming with the Message-Passing Interface.* Massachusetts Institute of Technology. Cambridge - USA. Third Printing, 1996.

Harvey, Charles A. e Gunter Stein (1978). Quadratic Weights for Asymptotic Regulator. *IEEE Transactions on Automatic Control* **23**(3), 378–387.

Holland, J.H. (1962). Outline for a logical theory of adaptive system theory. *Journal of the Association for Computing Machinery* **2**, 297–314.

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

Huntley, Christopher L. e Donald E. Brown (1996). Parallel Genetic Algorithms with Local Search. *Computers Ops Research - Elsevier Science Ltd* **23**(6), 559–571.

Kailath, Thomas (1980). *Linear Systems.* first ed.. Prentice-Hall Inc.. Englewood Cliffs, N.J.,USA.

Kalman, R. E. (1960*a*). On the General Theory of Control. *Proceedings of First IFAC Congress* **1**, 481–491. Butter Worth, London.

Kalman, R.E. (1960*b*). Contribution to the Theory of Optimal control. *Bol. Sociead Matematica Mexicana* **5**, 102–119.

Kalman, R.E. (1963). The theory of optimal control and the calculus of variations. *in Mathematical Optimization Techniques, R. E. Bellman, Ed., Berkeley CA, Universisty of California Press.*

Kalman, R.E., P. Falb e M.A Arbib (1969). *Topics in Mathematical System Theory.* McGraw-Hill. New York, USA.

Kalman, R.E., R. W. Koepcke e N. Y. Poughkeepsie (1958). Optimal Synthesis of Linear Sampling Control System Using Generalized Performance Indexes. *Transactions. ASME Ser. D (J. Basic Engeneering* **80**, 1820–1826.

Kawasaki, Naoya e Etsujiro Shimemura (1983). Determining Quadratic Weighting Matrices to Locate Poles in a Specified Region. *Automatica* **19**(5), 555–560.

Kimura, Hidenori (1975). Pole Assignment by Gain Output Feedback. *IEEE Transaction on Automatic Control* **20**(4), 509–516.

Klein, G e B.C. Moore (1976). Eigenvalue-Generalized Eigenvector Assignment with State Feedback. *IEEE Transaction on Automatic Control* **21**(6), 140–141.

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

Kristinsson, K. e G.A. Dumont (1992). System identification and control using genetic algorithms. *IEEE Trans. on Systems, Man and Cybernetics* **22**(5), 1033–1046.

Laub, Alan J. (1979). A Schur Method for Solving Algebraic Riccati Equations. *IEEE Transactions on Automatic Control* **24**(6), 913–921.

Letov, A. M. (1960). Analytic Controller Design I e II. *Automation and Remote Control* **21**, 303–306.

Littleboy, Darren M. e N. K. Nichols (1998). Modal coupling in linear control systems using robust eigenstructure assignment. *Linear Algebra and its Apllications* pp. 359–379.

Liu, G.P. e R.J Patton (1998). *Eigenstructure Assignment for Control System Design.* John Willey & Sons.

Luenberger, David G. (1979). *Introduction to Dynamic Systems.* John Willey & Sons, INc.

Mac Farlane, A. G. M (1974). *Relationships Between Recent Developments in Linear Control Theory and Classical Design Techniques the book - Control System Design by Pole-Zero Assignment - Edited by F. Falside.* Academic Press Inc.. London.

Manderick, Bernard e Piet Spiessens (1989). Fine-Grained Parallel Genetic Algorithms. *Proceedings of* 3^{th} *International Conferece on Genetic Algorithms* pp. 428–433.

Marra, M. A. e B. L. Walcott (1996). Stability and Optimality in Genetic Algorithm Controllers. In: *Proceedings of the IEEE international symposium of intelligent control.* Vol. 1. Dearborn, MI,USA. pp. 492–496.

Marrison, C.I e R.F Stengel (1997). Robust Control System Design Using Random Search and Genetic Algorithms. *IEEE Transaction on Automatic Control* **42**(6), 835–839.

Medanic, J., H.S. Tharp e W.R. Perkins (1988). Pole Placement by Performance Criterion Modification. *IEEE Transactions on Automatic Control* **33**(5), 469–472.

Mühlenbein, H. (1989). Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization. *Proceedings of* 3^{th} *International Conference on Genetic Algorithms* pp. 416–421.

Michalewicz, Zbigniew, Jacek B. Krawczyk, Mohammad Kazemi e Cezary Z. Janikw (1992). Genetic Algorithms and Optimal Control Problems. In: *Proceedings of the 29th Conference on Decision and Control- IEEE.* Honolulu, Hawaii- USA. pp. 1644–1666.

Moore, B.C. (1976). On the Flexibility Offered by State Feedback in Multivariable Systems Beyond Closed Loop Eigenvalue Assignment. *IEEE Transaction on Automatic Control* **21**(6), 689–692.

Morales, Daniel Santos Monasterios (1995). *Sintonia de Controladores DMC Utilizando algoritmos Genéticos.* Dissertação de Mestrado- Faculdade de Engenharia Elétrica e de Computação- UNICAMP. Campinas-São Paulo-Brasil.

Nichols, N.K. e P. Van Dooren (1984). Robust Pole Assignment and Optimal Stabillty Margins. *Eletronic Letters* **20**, 660–61.

Ochi, Y. e K. Kanai (1996). Eigenstructure Assignment for Linear Quadratic Regulator. *IFAC Proceedings of 13th Triennial World Congress - San Francisco - USA* pp. 115–120.

Patel, Rajnikant V. e Neil Munro (1982). *Multivariable System Theory and Design.* Pergamon Press.

Petkov, P. Hr., N. D. Christov e M. M. Konstantinov (1991). *Computational Methods for Linear Control Systems.* Prentice Hall International. UK.

Pettey, Chrisila C. e Michael R. Leuze (1989). A Theoretical Investigation of Parallel Genetic Algorithms. *Proceedings of* 3^{th} *International Conference on Genetic Algorithms* pp. 398–405.

Porter, B. (1969*a*). Assignment of closed-loop eigenvalues by the direct method of Lyapunov. *International Journal of Control* **10**(2), 153–157.

Porter, B. (1969*b*). Eigenvalue sensitivity of Modal control to Loop gain Variations. *International Journal of Control* **10**(2), 159–162.

Porter, B. e J.J. D'Azzo (1977). Algorithm for the Synthesis of State Feedback Regulator by Entire Eigenstructure Assignment. *Electronics Letters* **13**(8), 230–231.

Porter, L.L. e K. M. Passino (1994). Genetic model reference adaptive control. In: *Proc IEEE in Symp Intelligent.* Columbus-Ohio. pp. 219–224.

Press, William H., Saul A. Teukolsky, Willam T. Vetterlng e Brian P. Flannery (1994). *Numerical Recipes in Fortran: The Art of Scientific Computing.* second ed.. Cambridge Universisty Press. P.O. box243 - Cambridge, MA 022838 - USA.

Rissanen, J. (1960). Control Systems Synthesis by Analogue Computer Based on the Generalized Linear Feedback Concept. In: *Proceedings of the Symposium on Analog Computation Applied to the Study of Chemical Processes. International Seminar. Brussels, Belgium.* Vol. 1. pp. 1–13.

Rosenbrock, H. H. (1962). Distinctive Problems of Process Control. *Chem. Eng. Prog.* **58**, 43–50.

Rosenbrock, H. H. (1970). *State Space and Multivariable Theory.* Nelson, London.

Rosenbrock, H. H. (1974). *Computer-aided Control System Design.* Academic Press, London.

Sage, Andrew P (1968). *Optimum systems control.* Prentice-Hall, Inc. Englewood Cliffs.

Salukvadze, M.E. (1979). *Vector-Valued Optimization Problems in Control theory.* Academic Press. New York, NY.

Sobel, K. M. e E. Y. Shapiro (1985*a*). Eigenstructure Assignment: A Tutorial Part I Theory. In: *Proceedings of American Control Conference* 5^{th} . Vol. 1. Saint-Nazaire, USA. pp. 456–460.

Sobel, K.M. e E.Y. Shapiro (1985*b*). Eigenstructure Assignment: A Tutorial part II Applications. In: *Proceedings of American Control Conference* 5^{th} . Vol. 1. Saint-Nazaire, USA. pp. 461–462.

Steen, Aad J. van der (1997). *Overview of recent supercomputers.* Strinching Nationale Computer Faciliteiten. Netherlands.

Stein, G. (1979). Generalized Quadratic Weights for Asymptotic Regulator Properties. *IEEE Transactions on Automatic Control* **24**(4), 559–566.

Tamariz, Annabell D.R (1999). Uma nova proposta para solução computacional da equação algébrica de Riccati em formas sequencial e paralela. Master's thesis. Universidade Estadual de Campinas - UNICAMP. Campinas, SP, Brasil.

Tanese, Reiko (1989). Distributed Genetic Algorithms. *Proceedings of* 3^{th} *International Conference on Genetic Algorithms* pp. 429–439.

Whidborne, J.F, D.W. GU e I. Postlewaite (1997). Simulated Annealing for Multiobjective Control System Design. *IEE Proc-Control Theory Appl.* **144**(6), 582–588.

White, B.A. (1991). Assignment of Eigenstructure by use of Polynomial Matrices . *Journal of Systems and Control Engineering - Proc Instn Mech Engers.* **205**, 207–214.

White, B.A. (1995). Eigenstructure Assignment: a survey. *Journal of Systems and Control Engineering - Proc Instn Mech Engers.* **209**, 1–11.

Wiener, N. (1948). *Cybernetics.* Wiley - New york.

Wilkinson, J.G. (1965). *The Algebraic Eigenvalue Problem.* Oxford-Press . Oxford-UK.

Wonham, W. M. (1967). On Pole Assignment in Multi-Input Controllable Linear System. *IEEE Transactions on Automatic Control* **12**(6), 660–665.

Zadeh, L.A. (1963). Optimality and Uncertainty in Control System Design. *IEEE Transactions on Automatic Control* **AC-8**, 59–60.

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

Zargham, Mehdi R. (1995). *Computer Architecture: Single and Parallel Systems*. Prentice-Hall Inc.. Upper Saddle River - NJ - USA.

Índice Remissivo de Palavras-Chave

Índice Remissivo de Autores